НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

"___" _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою « Системне програмування »
спеціальності **123 «Комп'ютерна інженерія»**

на тему: HDL-модель пристрою корекції модульних помилок

Виконав:

студент IV курсу, групи КВ-62
Мельник Андрій Олександрович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник асистент Коляда К.В _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____2020 р.

ЗАВДАННЯ
на дипломний проєкт студента
Мельник Андрій Олександрович

1. Тема проєкту «HDL-модель пристрою корекції модульних помилок», керівник проєкту асистент Коляда Констянтин Вячеславович, затверджені наказом по університету від «25» травня 2020 р. № N1181-С

2. Термін подання студентом проєкту «___» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки
   - пошук способів виявлення і виправлення помилок у запам'ятовуючому пристрої
   - вивчення кодів корекції модульних помилок
   - створення HDL-моделі пристрою

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- Блок кодування. Блок-схема алгоритму
- Блок декодування. Блок-схема алгоритму
- Блок обчислення контрольних розрядів. Блок множення (MUL). Схема електрична структурна
- HDL-модель пристрою корекції модульних помилок. Схема електрична структурна
- Блок обчислення синдрому. Блок корекції визначеної помилки. Схема електрична структурна

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення літератури за тематикою проекту	10.02.2020	
2	Розроблення та узгодження технічного завдання	13.03.2020	
3	Аналіз існуючих рішень	18.03.2020	
4	Підготовка матеріалів першого розділу дипломного проєкту	30.03.2020	
5	Підготовка матеріалів другого розділу дипломного проєкту	10.04.2020	
6	Підготовка матеріалів третього розділу дипломного проєкту	25.04.2020	
8	Підготовка звіту дипломного проекту	10.05.2020	
9	Передзахист дипломного проекту	20.05.2020	

Студент _____ Андрій Мельник
(підпис)

Керівник проєкту _____ Констянтин Коляда
(підпис)

* Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

В ході виконання даного дипломного проєкту було розроблено HDL-модель, за допомогою якої можна провести аналіз поведінки пристрою корекції модульних помилок в тому числі: виконання зберігання даних, їх кодування і декодування, виправлення одиночних модульних помилок і також повідомляти про подвійні модульні помилки. Розроблена модель виконує кодування за допомогою подовженого модульного коду Хемінга. Також розроблена модель виконує декодування і виправлення закодованої інформаційної послідовності за синдромом. Під час розробки було поставлено ціль створення HDL-моделі з швидкою та правильною корекцією модульних помилок, і можливістю зберігати інформацію користувачеві без необхідності перевіряти інформацію.

В дипломному проєкті наведено основні теоретичні відомості щодо поняття кодування, декодування, виявлення і виправлення помилок у запам’ятовуючому пристрої. Зокрема, особлива увага в проєкті звертається на опис алгоритму декодуванні по синдрому.

У роботі було проведено аналіз результатів роботи змодельованої програми на основі різних тестових наборів,.

Всі компоненти програми розроблені мовою програмування AHDL. Даний вибір середовища розробки дозволяє моделювати програми і пристрої на їх базі, які можна буде імплементувати у ВІС.

Дипломний проєкт містить: 51 ст., 15 рис., 1 табл., 9 посилань на використаних джерел

ANNOTATION

During the implementation of this diploma project, an HDL model was developed, which can be used to analyze the behavior of the device modular error correction, including: storage, encoding and decoding, correction of single modular errors and also to report double modular errors. The developed model performs coding using the extended modular Heming code. The developed model also performs decoding and correction of the encoded information sequence according to the syndrome. During the development, the goal was to create an HDL-model with fast and correct correction of modular errors, and the ability to save information to the user without having to check the information.

The diploma project provides basic theoretical information on the concept of encoding, decoding, detection and correction of errors in the storage device. In particular, special attention in the project is paid to the description of the decoding algorithm for the syndrome.

The analysis of the results of the simulated program on the basis of different test sets was carried out in the work.

All components of the program are developed in the AHDL programming language. This choice of development environment allows you to model programs and devices based on them, which can be implemented in the BIS.

Diploma project contains: 51 articles, 15 figures, 1 table, 9 references to used sources

№ з/п	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	Примітки
1	А4	ІАЛЦ.467200.002 ТЗ	HDL-модель пристрою	4	
			корекції модульних		
			помилок		
			Технічне завдання		
2	А4	ІАЛЦ. 467200.003 ТП	HDL-модель пристрою	2	
			корекції модульних		
			помилок		
			Відомість технічного		
			проекту		
3	А4	ІАЛЦ. 467200.004 ПЗ	HDL-модель пристрою	50	
			корекції модульних		
			помилок		
			Пояснювальна записка		
4	А4	ІАЛЦ. 467200.005 Д1	HDL-модель пристрою	1	
			корекції модульних		
			помилок		
			Блок кодування		
			Блок-схема алгоритму		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.001 ОА		
Розробив		Мельник А.О.			HDL-модель пристрою корекції модульних помилок **Опис альбому**	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Коляда К.В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О					

№ з/п	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	Примітки
5	А4	ІАЛЦ. 467200.006 Д2	HDL-модель пристрою	1	
			корекції модульних		
			помилок		
			Блок декодування		
			Блок-схема алгоритму		
6	А4	ІАЛЦ. 467200.007 Е1	HDL-модель пристрою	1	
			корекції модульних		
			помилок		
			Схема електрична		
			структурна		
7	А4	ІАЛЦ. 467200.008 Е1	Блок обчислення	1	
			контрольних розрядів.		
			Блок множення (MUL).		
			Схема електрична		
			структурна		
8	А4	ІАЛЦ. 467200.009 Е1	Блок обчислення	1	
			визначеної помилки.		
			Схема електрична		
			структурна		

№ з/п	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	Примітки
9		Диск CD-ROM	Текст пояснювальної	1	
			записки.		
			Графічний матеріал		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.001 ОА	Арк. 3

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: HDL-модель програми корекції модульних помилок.

Галузь застосування: створення програми для зменшення кількості помилок при зберіганні інформації.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є розробка HDL-моделі пристрою корекції модульних помилок з подальшим моделюванням його роботи для аналізу поведінки в компю'терних системах з можливою появою помилок при збереженні інформації.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- можливість кодування вхідної інформації;
- можливість декодування закодованої вхідної модульної послідовності;

- можливість отримання інформації від пристроїв збереження інформації;

5.2 Вимоги до апаратного забезпечення

- оперативна пам’ять: 1 Гб;

5.3 Вимоги до програмного та апаратного забезпечення користувача

- операційна система Windows;
- програма для компіляції мови моделювання AHDL.

6. **ЕТАПИ РОЗРОБКИ**

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	10.02.2020
2.	Розроблення та узгодження технічного завдання	13.03.2020
3.	Аналіз існуючих рішень	18.03.2020
4.	Підготовка матеріалів розділів дипломного проекту	10.04.2020
5.	Підготовка звіту дипломного проекту	10.05.2020
6.	Передзахист дипломного проекту	20.05.2020

№ з/п	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	Примітки
1	А4	ІАЛЦ. 467200.004 ПЗ	HDL-модель пристрою	50	
			корекції модульних		
			помилок		
			Пояснювальна записка		
2	А4	ІАЛЦ. 467200.005 Д1	HDL-модель пристрою	1	
			корекції модульних		
			помилок		
			Блок кодування		
			Блок-схема алгоритму		
3	А4	ІАЛЦ. 467200.006 Д2	HDL-модель пристрою	1	
			корекції модульних		
			помилок		
			Блок декодування		
			Блок-схема алгоритму		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.003 ТП		
Розробив		Мельник А.О.			HDL-модель пристрою корекції модульних помилок **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Коляда К.В.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О					

№ з/п	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	Примітки
4	А4	ІАЛЦ. 467200.007 Е1	HDL-модель пристрою	1	
			корекції модульних		
			помилок		
			Схема електрична		
			структурна		
5	А4	ІАЛЦ. 467200.008 Е1	Блок обчислення	1	
			контрольних розрядів.		
			Блок множення (MUL).		
			Схема електрична		
			структурна		
6	А4	ІАЛЦ. 467200.009 Е1	Блок обчислення синдрому.	1	
			Блок обчислення		
			визначеної помилки.		
			Схема електрична		
			структурна		
7		Диск CD-ROM	Текст пояснювальної	1	
			записки.		
			Графічний матеріал		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.003 ТП	Арк. 2

Пояснювальна записка
до дипломного проєкту

на тему: HDL-модель пристрою корекції модульних помилок

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БАС – блок аналізу синдрома

БВП – блок визначення помилки

БД – блок декодування

БК – блок кодування

БКВП – блок корекції визначенної помилки

БОС – блок обчислення синдрому

БКР – блок обчислення контрольних розрядів

ВІС – велика інтегральна схема

ДС – дешифратор синдрома

ЗП – запам’ятовуючий пристрій

КБ – коригувальний блок

КХ – код Хеммінга

МП – мікросхема пам'яті

ОС – обчислювальна система

ПВВП – пристрій виявлення і виправлення помилок

ВСТУП

Напівпровідникові запам'ятовуючі пристрої (ЗП) знаходять широке застосування в обчислювальних і керуючих комплексах, системах обробки і зберігання інформації. Вони є найбільш важливими функціональними вузлами сучасної елементарної бази.

Розробка нових типів ВІС ЗП йде, в основному, в напрямку збільшення ступеня інтеграції і досягається зменшенням розмірів елементів і більш щільною їх компонуванням, що призводить до появи великої кількості випадкових дефектів при виробництві і щодо ПВВП інтенсивності відмов і збоїв в процесі експлуатації.

Технологічні методи підвищення виходу придатних і надійності ВІС ЗП в багатьох випадках виявляються не ефективними, тому що або вимагають великих капітальних витрат і пов'язані з тривалими попередніми дослідженнями, або взагалі обмежені існуючим рівнем науково-технічного прогресу. Тому для вирішення зазначеної проблеми в останні роки все наполегливіше стали залучатися методи, засновані на введенні в напівпровідникові ЗП надлишкових елементів, що дозволяють нейтралізувати дефекти виробництва, відмови і збої, що з'являються в процесі експлуатації.

Реалізація в системах пам'яті на ВІС ЗП блоків кодування і декодування навіть при виправленні одиночних помилок вимагає використання великої кількості мікросхем середнього ступеня інтеграції, що не завжди допустимо через ускладнення пристрою, зниження швидкодії, збільшення маси і габаритів.

Крім того, розробка блоків корекції для кожної конкретної системи пов'язана зі значними економічними витратами і негативно позначається на вартості продукції, що випускається. У зв'язку з цим виникає питання про розробку пристроїв виявлення і виправлення помилок для напівпровідникової пам'яті у вигляді ВІС, орієнтованої на досить широке використання. Це дозволить застосовувати пристрої виявлення та виправлення помилок (ПВВП) не тільки у великих ЕОМ, промислових контролерах, системах військового і комерційного призначення, а й в мікропроцесорних пристроях.

1. ВИЯВЛЕННЯ І ВИПРАВЛЕННЯ ПОМИЛОК У ЗП

1.1 Основні вимоги для побудови ПВВП

Для массового використання користувачами ВІС ПВВП мають дотримуватися деяким вимогам. Такі вимоги показують основні функції, які необхідні для правильної і швидкої роботи ВІС. Деякі з цих вимог:

1. Найбільша кількість знайдених помилок при найменш можливою кількістю додаткових перевірочних розрядів.

2. Висока швадкість обробки інформації.

3. Невисока складність ВІС ПВВП.

4. Робота з різними форматами довжин слів шляхом нарощування числа ВІС ПВВП при збільшенні оброблюваної довжини слова понад використовуваної в ВІС ПВВП і по можливості без залучення додаткових мікросхем малої і середньої ступенів інтеграції.

5. Забезпечення роботи систем з ПВВП в різних режимах контролю пам'яті (апаратно-програмному, апаратному, прямому пропущенні без корекції помилок, діагностичному, побайтно запису інформації).

6. Мітки помилок для виявлення однократної і багатократної помилок.

7. Мітки від розряди регістра синдрома, що дозволяє здійснювати нарощування числа ВІС ПВВП, а також збільшує функціональні можливості систем по корекції помилок.

8. Мітки для виводу режиму роботи і формату оброблюваного слова.

9. Мультиплексований введення-виведення даних та перевірочних розрядів, що зменщує кількість введення-виведення даних.

1.2 Вибір коригуючого коду для побудови ПВВП

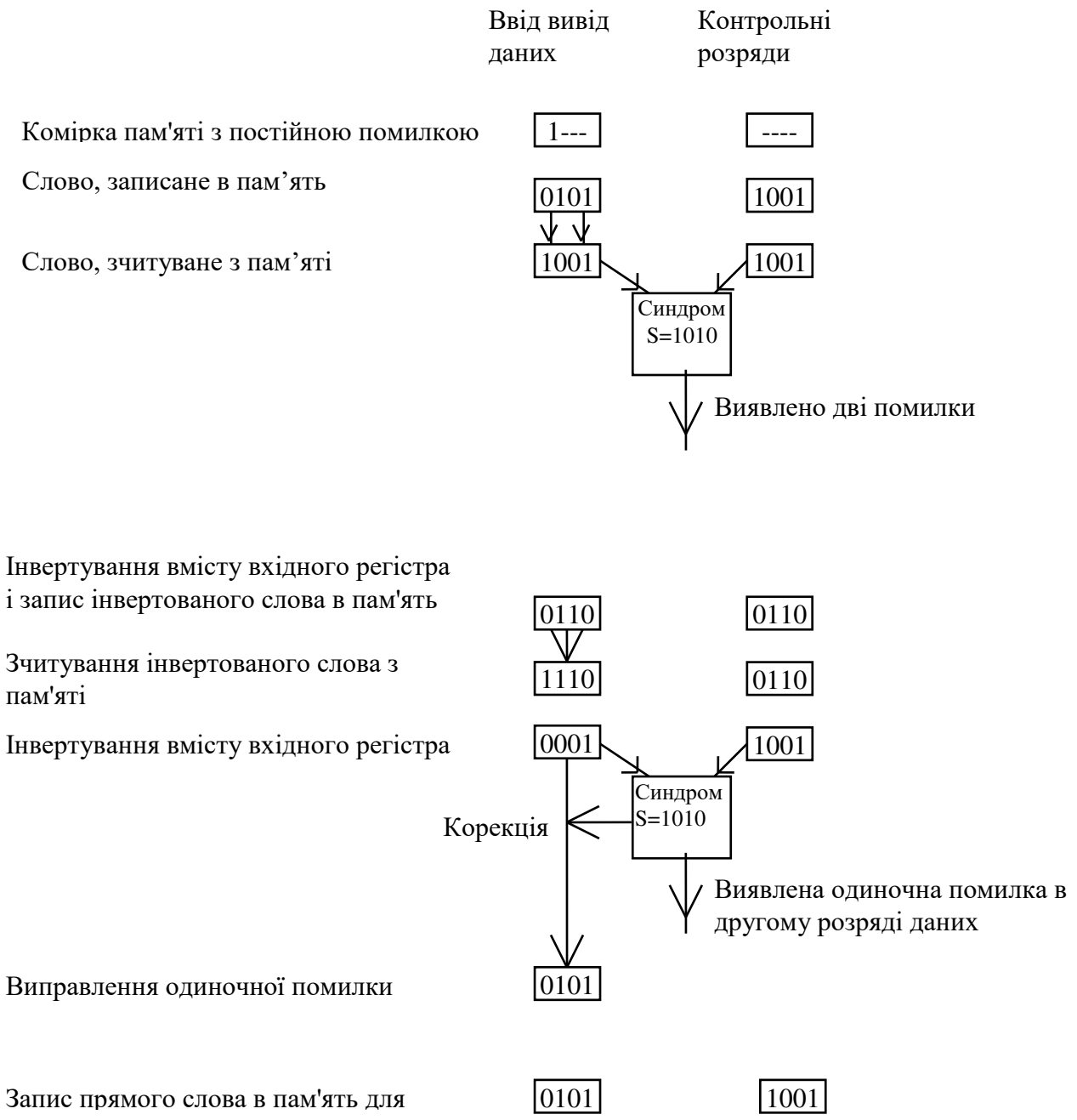

Рис. 1.1. Послідовність корекції подвійних помилок за допомогою подвійного інтегрування

$$\mathbf{G} = \begin{bmatrix} 1000 & 1101 \\ 0100 & 0111 \\ 0010 & 1011 \\ 1111 & 1111 \end{bmatrix}. \qquad (1.1)$$

Вибір коду для побудови ВІС ПВВП з корекцією подвійних помилок пов'язаний з певними труднощами в наслідок обмежень на число перевірочних розрядів і складність декодуючого блоку. Відомі коди для виправлення подвійних помилок мають велику надмірність і досить складні схеми декодування. Разом з тим подвійні помилки можуть бути виправлені з допомогою подовжених кодами Хеммінга. В режимі корекції подвійних помилок при появі ненульового синдрому парного ваги (що вказує на подвійну помилку) лічену слово інвертується і записується назад в пам'ять за тією ж адресою. Потім проводиться зчитування інвертованих даних з пам'яті, повторне інвертування та їх корекція. При цьому відбувається відновлення вихідних даних з виправленням помилок в наслідок відмов елементів пам'яті.

Можливі наступні ситуації: відмова двох елементів пам'яті (помилки в двічі інвертованому слові відсутні); відмова одного елемента і збої іншого (двічі інвертованому слові залишається тільки помилка на місці збою, виправлена кодом); збій двох елементів (помилки такого роду не виправляються, так як призводять до спотворення двох позицій і двічі інвертованому слові).

На рис. 1.1 наведено приклад корекції подвійних помилок зазначеним методом за допомогою коду (8; 4), що задається матрицею, що породжує G (1.1). можливий і дещо інший спосіб корекції подвійних помилок. При повторному прочитуванні інвертованого слова з пам'яті відбувається його порозрядне порівняння з прямим словом, зберігаються в регістрі. Той розряд, де відбулося збіг, відзначається як відмовив, оскільки стан відмовив розряду не залежить від записуваної інформації; далі проводиться інвертування цього розряду в прямому слові і видача результату в ПВВП для виправлення можливої помилки в наслідок збою елемента пам'яті.

Обидва цих способу корекції подвійних помилок мають серйозний недолік - низька швидкодія в наслідок необхідності після виправлення помилок повторно записувати в пам'ять пряме слово. В іншому разі в пам'яті буде зберігатися ніяк не зазначене інвертоване слово. При повторному зверненні до

прямого речі для виправлення подвійних помилок знову буде потрібно два цикли запису-зчитування. Крім того, оскільки помилки в наслідок постійних відмов залишаються, то поява додаткової помилки, наприклад в наслідок збою, призведе до невиправних помилок зберігання, що знижує надійність зберігання. Звідси випливає, що якщо позначити якимось чином збережене в пам'яті інвертоване слово і домогтися того, щоб воно належало кодом, можна збільшити швидкодію і надійність пам'яті з ПВВП.

Це досягається шляхом спільного використання коду, що виправляє одиночні помилки і поодинокі дефекти. Характерною рисою такого коду є наявність в ньому як прямих, так і інверсних слів коду, що виправляє одиночні помилки. Крім того, в одному з інформаційних розрядів містяться відомості про покриває векторі, нульовий символ в цьому розряді говорить про те, що слово зберігається в прямому коді, а одиничний - в інверсному. Слід зазначити, що цей розряд також захищений кодом від помилок. Таким чином, немає необхідності вдруге інвертувати прочитане інверсне слово і записувати його в пам'ять, що збільшує швидкодію на цикл запису, але зменшує на одиницю число інформаційних символів. З іншого боку, якщо на швидкодію УІОІ не накладаються жорстких обмежень і більш важливим є забезпечення надійності, то після виправлення помилки через збій скориговане інверсне слово знову слід занести в пам'ять. Це слово вже не містить помилок, що збільшує надійність в наступних циклах роботи.

Розглянемо з цієї точки зору коди Хеммінга з d = 3; 4. У кодах з d = 3 синдроми одиночних і подвійних помилок невиразні. Тому при кожному виявленні помилок необхідно інвертувати кодове слово і записувати його в пам'ять . Це призводить до зниження швидкодії при корекції одиночних помилок. У кодах d = 4 синдроми одиночних помилок мають одиночний вага, а синдроми подвійних парний. Тому виправлення одиночної помилки можна виробляти без інвертування слова і записи його в пам'ять; інвертування здійснюється тільки при виявленні подвійної помилки. На рис. 1.2 наведено приклад послідовної обробки даних (захищених кодом (8; 4) задається

породжує матрицею G(1.1)), четвертий інформаційний заряд яких відведено під перевірки розряд коду, що виправляє одиночні дефекти:

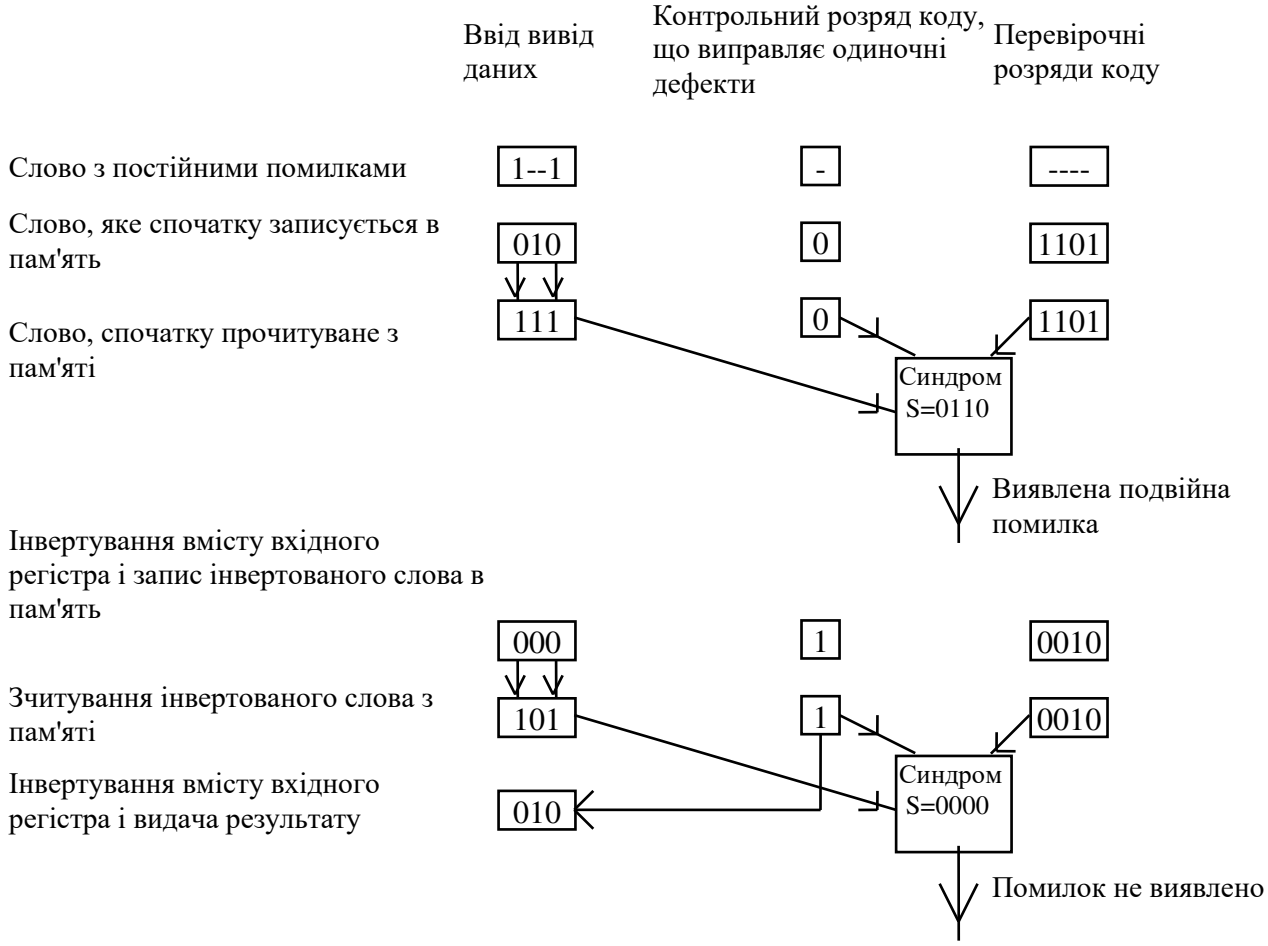

Рис. 1.2 Послідовність корекції подвійних помилок за допомогою коду, що виправляє одиночні помилки і дефекти

Таким чином, з урахуванням того, що один з інформаційних символів відводиться під перевірочну позицію коду, що виправляють поодинокі дефекти, необхідне число інформаційних позицій слід збільшити на одиницю. Це досягається укорочением довшого коду. Для забезпечення транспаратна довжина коду повинна бути парному. парність довжини забезпечується вибором коду з непарним числом перевірочних позицій.

В табл. 1.1 наведені параметри укорочених транспарантних кодів з d = 4.

В табл. 1.1 показується, що підвищення швидкодії за рахунок виключення циклу записи прямого слова (після корекції) і циклу записи

інверсного слова (при одиночних помилках) призводить до збільшення на один або два числа перевірочних розрядів.

Число розрядів даних *k*	16	32	48	64	80	128
Число розрядів слова *n*	24	40	56	74	90	138
Число контрольних розрядів *r*	8	8	8	10	10	10
Вибраний для скорочення код	(64;57)	(128;120)	(128;120)	(256;247)	(256;247)	(256;247)
Число контрольних разрядов нетранспарантних кодів с *d* = 4	6	7	7	8	8	9

Таблиця 1.1. Параметри укорочених транспарантних кодів з d = 4

1.3 Аналіз параметрів ПВВП

Більшість ПВВП працюють в чотирьох основних режимах: виявлення і виправлення помилок, обчислення контрольних розрядів, виявлення помилок, прямий пропуск в режимі зчитування. Також існують ПВВП, які працюють з трьома додатковими режимами, які використовуються для додаванні ПВВП додаткових властивостей во швидкодії, надійності і кількості контрольних розрядів.

В режимі виявлення і виправлення помилок ПВВП використовується для виявлення помилок в контрольному слові і корекції їх при кожному зверненні до пам'яті. При цьому структура системи спрощується, але збільшується час на считування даних.

В режимі обчислення контрольних розрядів відбувається обчислення контрольних розрядів по даним на вхідному регістрі даних ПВВП. Після цього відбувається видача даних і перевірочних розрядів для запису в пам’ять. Видача даних може відбуватися безпосереднь із ПВВП або із вихідних регістрів.

В режимі виявлення помилок ПВВП виконує тільки контрольні функції. Для його забезпечення ПВВП має виводи під мітки помилок. Виявляються помилки і індентифікуються на можливість виправлення шляхом виставлення міток помилок, але дані залишаються без змін. Завдяки цьому є можливість в кожний цикл обробки включити стан очікування і контролювати дані без суттєвого зменшення швидкості. Я кщо помилку не знайдено, тоді согнал очікування перестає діяти і цикл зчитування продовжується. Якщо помилка знаходится, тоді її виправленням займається ПВВП при блокуванні даних зчитаних з пам’яті, або програмним шляхом за допомогою центрального процесора.

В режимі прямого пропуску ПВВП забезпечує проходження даних і перевірчих розрядів без змін від входу до виходу.

Також можливі додаткові режими роботи ПВВП. Наприклад, режим діагностики. Цей режим задається контролюючими сигналами для вибору необхідних діагностичних операцій. Також режим діагностичних операцій містить в собі діагностичний регістр. За допомогою цього режиму є можливість переведення пам’яті ПВВП в нульовий стан і початкової установки вхідного регістру при одночасному формуванні контрольних розрядів і запис даних в пам’ять.

У можливості діагностичного режиму входить контроль пам’яті і контроль ПВВП програмними засобами. Для цього в діагностичний регістр ПВВП записуються завчасно відомі біти контролю, які використовуються при діагностичному зчитуванні та запису. Використовуючи різні комбінації діагностичних контролюючих розрядів і заносячи їх в пам’ять, можна виявити помилки в послідовному зчитуванні.

Відмінною особливістю всіх ПВВП є наявність на виходах пристроїв розрядів синдрому.

В усіх ПВВП є побайтний запис інформації, а також на виходах з пристрій розрядів синдрому. При побайтному записі інформації відбувається зменшення швидкості виконання процедури. Тому її використовують в операціїї попереднього зчитування з записом для програми тільки в одному циклі зчитування – модифікації запису. При цьому цикл збільшується на один такт.

1.4 Призначення мови HDL і принципи його побудови

Мова HDL призначений для опису проектів різного ступеня складності - від найпростішого вентиля до цілої системи, що складається з апаратних і програмних частин. Він дозволяє будувати моделі на різних рівнях абстракції, виконувати імітаційне моделювання і генерувати тимчасові діаграми, вести суворе документування проекту, здійснювати синтез структури по поведінковому опису, верифікувати проект формальними методами, автоматично генерувати тести.

Перші HDL були створенні в 1960-х на базі мови програмування ADA і успадковує багато його властивості. Додатково в мову введено поняття модельного часу, сигналу, події, компонента та інші. Використання HDL дозволяє не прив'язувати проект заздалегідь до конкретного фізичного способу реалізації, одна і та ж логічна HDL-реалізація є джерелом генерації різних фізичних.

В основі мови лежать наступні принципи побудови програми:

- підтримка функціональної декомпозиції (функціональна ієрархія і рекурсія);
- підтримка структурної декомпозиції (структурна ієрархія);
- представлення системи у вигляді паралельно функціонуючих взаємодіючих процесів;
- використання абстрактних типів даних;
- використання подієвого моделювання;

- підтримка різних рівнів абстракції та деталізації представлення проекту.

Відповідно до рівнів абстракції, мова допускає побудову моделей трьох типів:

- поведінкових;
- потокових;
- структурних.

Для побудови кожного типу моделей використовується своя характерна підмножина засобів мови.

Основні засоби мови HDL включають:

- бібліотеки;
- модулі: інтерфейси об'єкта проекта, архітектурні тіла, конфігурації, пакети, тіла пакетів;
- підпрограми: процедури, функції;
- скалярні типи даних: перераховуючі, числові, фізичні;
- складені типи даних: записи, масиви, файли;
- об'єкты: константи, змінні, сигнали, порти та інше;
- операції: логічні, порівняння, арифметичні;
- вирази;
- послідовні оператори: очікування, підтвердження, призначення сигналу, призначення змінної, виклик процедури, умовний, селективний, циклу, повернення та інше;
- паралельні оператори: блока, процесу, паралельного виклику процедури, паралельного підтвердження, паралельного призначення сигналу, конкретизації компонента, генерації.

В мові HDL є деякі особливості:

- Мова HDL не розрізняє великі та малі літери. Однак, рекомендується для поліпшення читаності ключових слів використовувати великі літери;

- Новий рядок також можна починати в будь-якому вільному місці, тобто на місцях порожніх рядків, табуляції, пробілів;
- Основні конструкції мови відокремлюються порожнім простором;
- Ключові слова, імена і числа повинні розділятися відповідними символами або операторами і/або одним або більше пробілами;
- Коментарі повинні бути укладені в символи відсотка (%). Коментарі не можуть бути вкладеними.

1.5 Опис HDL компонентів

Мова легко сприймається як машиною, так і людиною. Вона може використовуватися на етапах проектування, верифікації, синтезу і тестування апаратури, також для передачі даних про проект, модифікації і супроводу. Найбільш універсальним і поширеним мовою опису апаратури є HDL. Цією мовою можливо як поведінковий, так структурний і потоковий опис цифрових схем.

Мова HDL використовується в багатьох системах для моделювання цифрових схем, проектування програмованих логічних інтегральних мікросхем, базових матричних кристалів, замовних інтегральних мікросхем.

З точки зору програміста мова HDL складається ніби з двох компонент - загально арифметичного і проблемно-орієнтованого.

Загально арифметичний компонент HDL - це мова, близька по синтаксису і семантиці до сучасних мов програмування типу Паскаль, С та ін. Мова належить до класу строго типізованих. Крім вбудованих (пакет STANDART) скалярних типів даних (цілий, речовинний, булевский, бітовий, даних типу час, даних типу посилання) користувач може вводити свої типи даних (перелічувальний, діапазонний та ін.).

Крім скалярних даних можна використовувати агрегати: масиви array, в тому числі і бітові вектори bit_vector, і символьні рядки string, записи record, файли file.

Послідовно виконуються (послідовні) оператори HDL можуть використовуватися в описі процесів, процедур і функцій. До них належать:

- оператор присвоювання змінної (:=);
- послідовний оператор призначення сигналу (<=);
- послідовний оператор затвердження (assert);
- умовний (if);
- вибору (case);
- циклу (loop);
- порожній оператор (null);
- оператор повернення процедури - функції (return);
- оператор послідовного виклику процедури.

Фрагменти описів, які можуть незалежно аналізуватися компілятором і при відсутності помилок поміщатися в бібліотеку проекту (робочу бібліотеку Work), називаються проектними пакетами design unit. Такими пакетами можуть бути оголошення інтерфейсу об'єкта проекту entity, оголошення архітектури architecture, оголошення конфігурації configuration, оголошення інтерфейсу пакета package і оголошення тіла пакету package body.

Модулі проекту, в свою чергу, можна розбити на дві категорії: первинні і вторинні. До первинних пакетам відносяться оголошення пакета, об'єкта проекту, конфігурації. До вторинних - оголошення архітектури, тіла пакета. Один або кілька модулів проекту можуть бути поміщені в один файл, який називається файлом проекту design file.

Кожен проаналізований модуль проекту поміщається в бібліотеку проекту design library і стає бібліотечним модулем library unit.

Кожна бібліотека проекту в мові HDL має логічне ім'я (ідентифікатор).

По відношенню до сеансу роботи з HDL-системою існує два класи робочих бібліотек проекту: робочі бібліотеки і бібліотеки ресурсів.

Робоча бібліотека - це бібліотека WORK, з якої в даному сеансі працює користувач і в яку поміщається пакет, отриманий в результаті аналізу пакету проекту.

Бібліотека ресурсів - це бібліотека, яка містить бібліотечні модулі, посилання на які є в аналізованому модулі проекту.

У кожен конкретний момент часу користувач працює з однієї робочої бібліотекою і будь-якою кількістю бібліотек ресурсів.

Проблемно-орієнтована компонента дозволяє описувати цифрові системи в звичних, розробнику поняттях і термінах. До них належить:

- поняття модульного часу now;
- дані типу time, що дозволяють вказувати час затримки в фізичних одиницях;
- дані типу сигнал signal, значення яких змінюється не зразу, а з вказаною затримкою, а також спеціальні операції і функції над ними;
- засоби оголошення об’єктів entity і їх архітектур architecture.

Якщо говорити про операційну частину проблемно орієнтованої компоненти, то умовно її можна розділити на засоби поведінкового опису апаратури (паралельні процеси і засоби їх взаємодії), засоби потокового опису (опис на рівні міжрегістрових передач) - паралельні оператори призначення сигналу (<=) з транспортною transport або інерційною затримкою передачі сигналів і засоби структурного опису об'єктів (оператори конкретизації компонент із завданням карт портів port map і карт настройки generic map, оголошення конфігурації і т.д.). До паралельних операторів HDL входять:

- оператор процесу process;
- оператор блока block;
- паралельний оператор прнзначення сигналу <=;
- оператор умовного призначення сигналу when;
- оператор селективного призначення сигналу select;
- паралельний оператор затвердження assert;
- паралельний оператор виклику процедури;
- оператор конкретизації компоненти port map;
- оператор генерації конкретизації generate;

Як видно з цього переліку, послідовні і паралельні операції призначення, виклику процедури і затвердження розрізняються контекстно, тобто всередині процесів і процедур вони послідовні, непаралельні.

Базовим елементом опису систем на мові HDL є блок. Блок містить розділ описів даних і розділ паралельно виконуваних операторів. Окремим випадком блоку є опис архітектури об'єкта. В рамках опису архітектури можуть використовуватися внутрішні, вкладені блоки. Поряд з усіма перевагами блокової структури програми і її відповідності природному ієрархічному поданням структури проекту оператори блоку мови HDL дозволяють встановлювати, умови заборони входу в блок. Тільки при істинності значення вираження заборони управління передається в блок і ініціює виконання операторів його тіла.

Спочатку HDL створювався як мова опису цифрових (digital) схем, в останні роки виконано велику роботу по розширенню мови на опис і моделювання аналогових (analogue) схем. Ухвалення цифро-аналогового HDL як стандарт додасть мови принципово нові можливості, збільшить його популярність, розширить сфери застосування.

Числа використовуються для подання констант в булевих виразах і рівняннях. Мова HDL підтримує всі комбінації десяткових, двійкових, вісімкових і шістнадцяткових чисел. Десяткові, двійкові, вісімкові і шістнадцяткові числа можна використовувати в будь-якій комбінації. Синтаксис запису чисел в мові HDL для кожної із зазначених систем числення наступний:

- десяткова - містить послідовність цифр 0 – 9;
- двійкова - містить послідовність з цифр 0, 1 і символу X (даний символ відображає байдужий стан), розміщених в лапках (""), наступних за буквою B;
- вісімкова - містить послідовність цифр від 0 до 7, розміщених в лапках, наступних за буквою O або Q;

- шістнадцяткова - містить послідовність цифр від 0 до 9 і букв від A до F, розміщених в лапках, наступних за буквою H або X.

Константи зручно використовувати як імена чисел. Перевага використання констант полягає в тому, що якщо одне і те ж число задано через ім'я константи і використовується в програмі кілька разів, то його значення можна змінити у всіх елементах програми, змінивши вказане значення тільки один раз при оголошенні константи. Перевага використання констант особливо помітно, якщо одне і те ж число використовується в файлі кілька разів. Тоді, якщо його потрібно змінити, змінюють його тільки один раз в оголошенні константи.

Ключові слова в мові HDL використовуються для позначення наперед зазначених констант і для позначення початку, кінця і переходів в оголошеннях чисел, операторів і констант.

Ключові слова можна використовувати як символічні імена, тільки якщо вони укладені в символи одинарних лапках ('). Їх можна також використовувати в коментарях.

Булеві вирази складаються з операндів, розділених логічними і арифметичними операторами і компараторами і додатково згруповані за допомогою круглих дужок. Вирази використовуються в булевих рівняннях також як і в інших операторах таких як Case і If Then.

Булевий вираз може бути одним з наступних:

1. Операнд (a, b [5..1], 7, VCC);
2. Підставлене посилання на логічну функцію (out [15..0] = 16dmux (q [3..0]));
3. Префіксний оператор (! Або -), який застосовується до булеву висловом (!C);
4. Два булевих вирази, розділених бінарним оператором (d1 $ d3);
5. Булевий вираз, укладений в круглі дужки (! Foo & bar).

Ви можете назвати булеві оператори та компаратори в файлах HDL для полегшення введення присвоювання ресурсів і для інтерпретації розділу Equations в файлі звіту.

1.6 Об`єкти і типи даних HDL

Об'єкти це область зберігання даних певного типу. Створюються об'єкти за допомогою декларації об'єкта, наприклад: variable number: INTEGER;

В результаті породжується об'єкт з ім'ям number, який зберігає целочисленную величину. Крім того, number декларується як клас variable.

Об'єкти - дані можуть бути трьох класів:

- constant (константа) - може зберігати окреме значення певного типу. Це значення визначається об'єкту на початку моделювання і не може змінюватися в процесі моделювання;
- variable (змінна) - об'єкт цього класу може зберігати окреме значення певного типу, проте, в процесі моделювання йому можуть присвоюватися різні значення. Для цього використовуються вирази присвоювання (variable assignment statement);
- signal (сигнал) - об'єкт даного класу має попереднє значення, має поточне значення і набір наступних значень.

Об'єкти класу signal моделюють дротяні з'єднання в схемах, в той час як змінні (variable) і константи (constant) використовуються для моделювання поведінки схеми, вони аналогічні об'єктів, що використовуються в мовах програмування C і Pascal.

Кожен об'єкт в мові HDL може зберігати значення, що відносяться до певного набору. Це безліч значень декларується за допомогою оголошення типу (type declaration). Тип - це ім'я, яке зв'язується з певним набором значень і набором операцій. Деякі типи зумовлені мовою HDL. Наприклад, BOOLEAN має набір значень FALSE, TRUE і набір операторів: and, or, nor, nand, not. У мові є можливість створювати нові типи з використанням декларацій і завдання набору операцій.

Всі можливі типи в HDL розпадаються на чотири великі категорії:

- scalar (скалярні);
- composite (композитні) - вони складаються з елементів одного типу (масиви) або різного типу (записи);
- access type (типи доступу) - забезпечують доступ до даного типу через покажчики;
- file types (тип - файл) - забезпечує доступ до об'єктів, що містить послідовності значень даного типу.

У свою чергу скалярні типи поділяються на чотири види:

- enumeration (перелічувальний тип);
- integer (цілий тип);
- physical (фізичний тип);
- floating point (тип "з плаваючою комою").

1.7 Особливості синтезу схем за описами на мові HDL

Між процесами моделювання (імітації поведінки схеми) синтезу схем з використанням мови HDL є істотні відмінності. Розглянемо деякі з них. На відміну від системи моделювання система проектування перетворює проектну інформацію в заданий формат. Серед них виділяються широко поширений формат EDIF, спеціалізовані формати різних фірм-виробників мікросхем, а також мови Verilog, HDL і інші. Формат EDIF є своєрідним стандартом де-факто, він є в більшості сучасних систем проектування і може використовуватися для обміну інформацією. Серед спеціалізованих форматів можна виділити формат XNF фірми Xilinx, який широко використовувався в ранніх версіях САПР цієї фірми. При перетворенні вихідного модуля в проміжний текст на мові HDL зазвичай здійснюється перетворення в структурні конструкції цієї мови, в яких використовуються схемотехнічні особливості використовуваної елементної бази. Система моделювання зазвичай використовує всі конструкції мови HDL, в той час, як система проектування використовує не всі його можливості. Зазвичай при синтезі не підтримуються

операції над типом Real (в цьому випадку при синтезі видається помилка), ігноруються ключове слово After, що не підтримується також ряд інших другорядних конструкцій мови. У системах проектування атрибут event може використовуватися тільки для вказівки фронтів синхросигналов в умовних операторах, переважно в операторах if.

У сучасних системах проектування для опису схем використовується тип std_logic, який замінює тип bit. Тип std_logic має наступні значення:

- 0 - логічний нуль;
- 1 - логічна одиниця;
- U - значення не ініціалізувати;
- X - невідоме значення;
- Z - високий вихідний опір;
- W - невідоме значення при слабкому джерелі сигналу;
- L - логічний нуль при слабкому джерелі сигналу;
- H - логічна одиниця при слабкому джерелі сигналу;
- `-`- невизначене значення.

1.8 Мікросхеми для виявлення помилок

Виявлення і виправлення помилок в напівпровідникової пам'яті за допомогою спеціалізованих ВІС знайшло досить широке застосування. У всіх відомих ВІС ПВВП використовується подовжений код Хеммінга, що

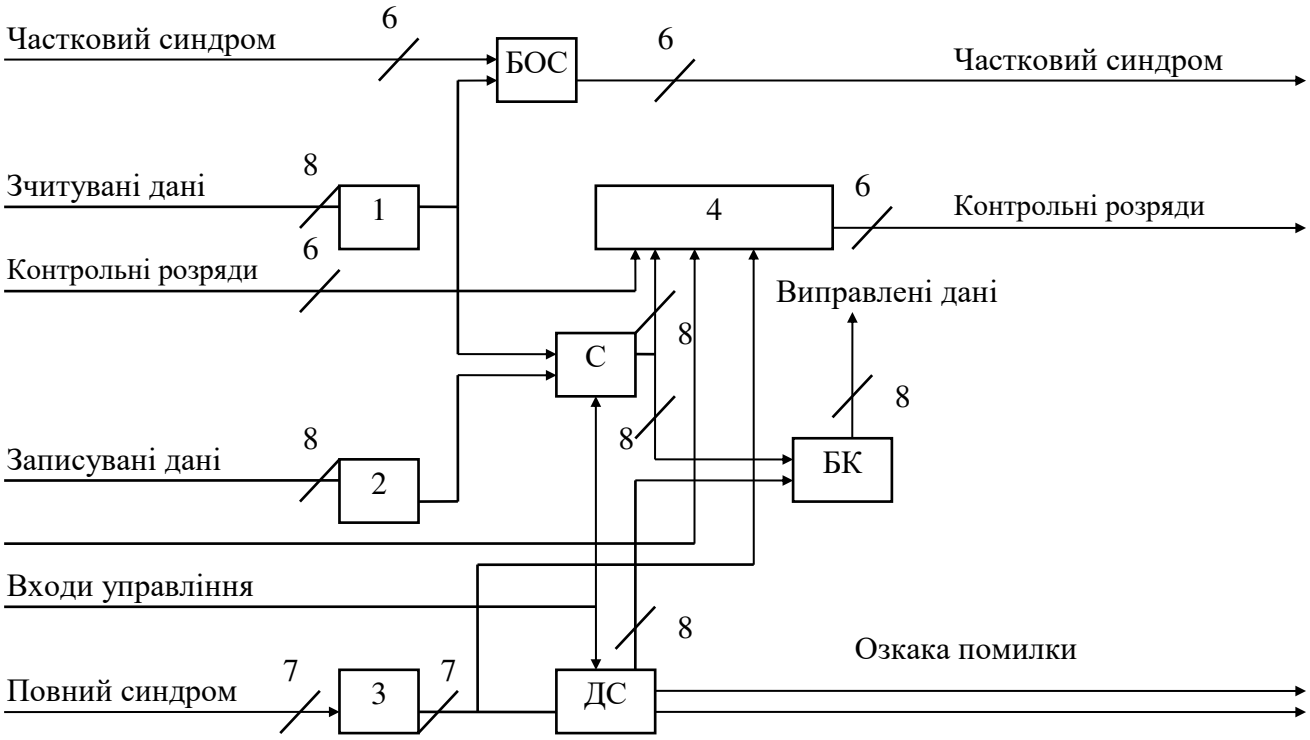

Рис. 1.3. Структурна схема нарощуваною 8-розрядної ВІС ПВВП: 1-буфер зчитувальних даних; 2-буфер записуваних даних; 3-буфер синдромних розрядів; 4-буфер обчислення і корекції перевірочних розрядів

забезпечує роботу з 8-, 16- або 32-розрядними даними, більшість ВІС передбачає нарощування секцій ПВВП для збільшення розрядності оброблюваного слова

На рис 1.3 описується ВІС ПВВП, що представляє собою 8-розрядну секцію для корекції одиночних помилок. На базі таких секцій можуть будуватися пристрою більшої розрядності. Кожна секція випускається в 64-вивідному корпусі. Пристрій має окремі вводи-виводи майже для всіх функцій, в тому числі нарощувані вводи синдромних і перевірочних розрядів, окремі вводи-виводи для байтів зчитування і запису даного для кінцевої секції (при нарощуванні 8-розрядних секцій ПВВП для обробки більше ніж 8-розрядних даних). Передбачені висновки для синдромних і перевірочних розрядів і виправленого даного. Велика кількість висновків робить пристрій гнучким, але вимагає застосування зовнішніх ІС середнього ступеня інтеграції для

мультиплексування і проміжного запам'ятовування розрядів даного. Крім того, потрібні ІС малому ступені інтеграції для обчислення прапорів-ознак одиночних

і дворазових помилок; ці прапори підлягають використанню процесором системи. Розглядається пристрій має тільки два робочих режиму (обчислення перевірочних розрядів в режимі запису і виявлення дворазових і виправлення одиночних помилок в режимі зчитування), перехід в які здійснюється за сигналом на лінії напряму. Тому управління ПВВП досить просто. З іншого боку, маршрутизація даних досить складна, особливо в 64-розрядних модифікаціях.

На рис. 1.4. представлено ВІС ПВВП , що виготовляється в двох різновидах: на елементи з трьома станами і на елементи з відкритим колектором. Пристрій орієнтований на застосування з 16 - і 32 - розрядними мікропроцесорами . Запис байта вимагає використання зовнішніх фіксаторів та інших ІС середньому ступені інтеграції , а також порівняно великих витрат машинного часу на управління з боку процесора . Зазначена операція може здійснюватися у режимі "читання – модифікація - запис": в напів "зчитування" відбувається вибірка інформації з ЗУ та її корекція при виявленні одиночної помилки , а в напів "модифікація-запис" запис байта і формування нових перевірочних розрядів. Подібно попередньому ВІС ПВВП розглянуте пристрій працює лише в двох режимах : обчислення перевірочних розрядів та виявлення і виправлення одиночних помилок . Однак останній режим здійснюється в три етапи : зчитування даних і перевірочних розрядів ,запам'ятовування і індикація

помилок шляхом встановлення прапорів , корекція даних – обчислення синдрому.

Пристрої випускаються в 28-,48-, 52-вивідних корпусах і не передбачає нарощування секцій для збільшення довжини оброблюваного слова . Два керуючих входу використовуються для встановлення режиму роботи , два інші

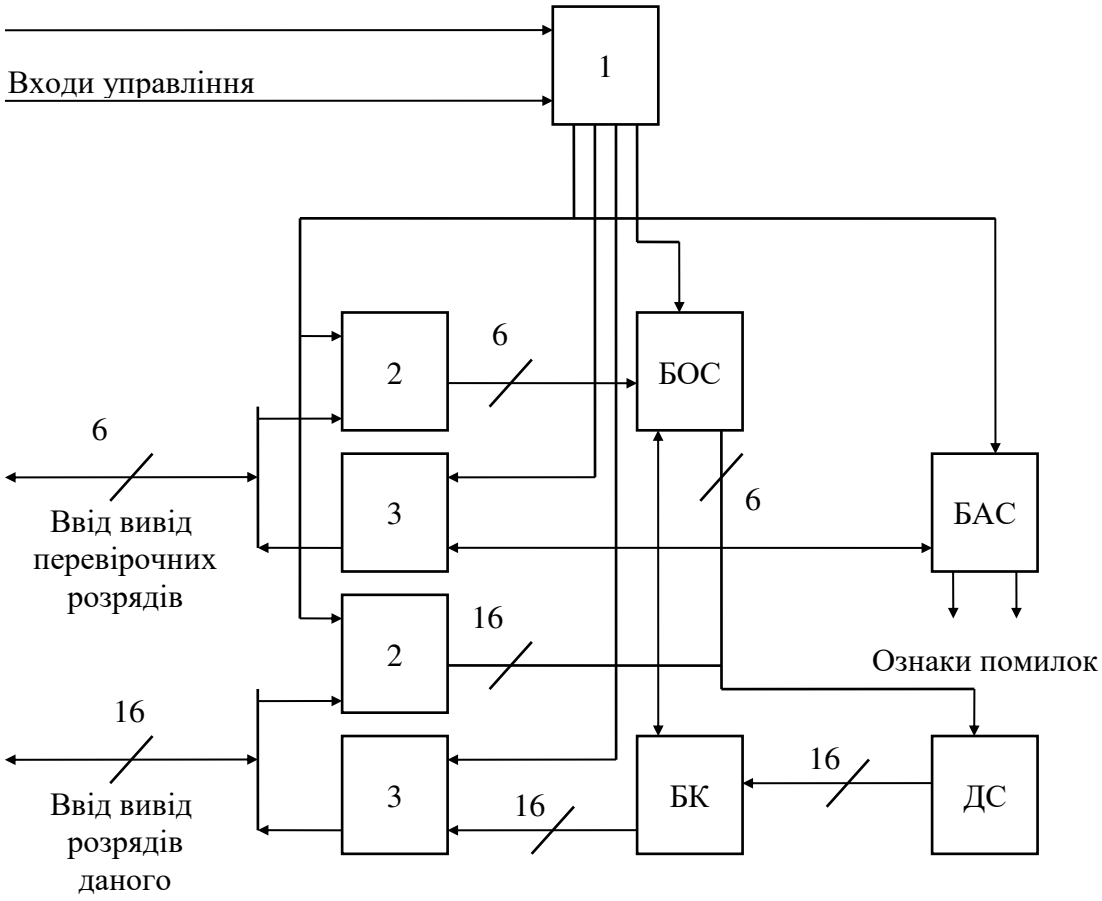

Рис. 1.4. Структурна схема не нарощуваною 16-розрядної ВІС ПВВП: 1 - блок управління; 2 - регістр; 3 - буфер

–для прапорів помилок (однократних та двократних), інші входи застосовують як двонаправлені шини передачі даних, перевірочних і синдромных розрядів. Враховуючи ,що більшість користувачів не важлива корекція контрольних розрядів , у пристрої передбачена сигналізація цього типу помилок (у формі встановлення прапорів) без обчислення нових перевірочних розрядів. Якщо бажано зберегти скориговані перевірочні розряди ,то необхідно повернути пристрій в режим обчислення перевірочних розрядів . Пристрій встановлює ознаки про виявлення двократних помилок ,але ці помилки не коригує . Крім

того , вона виявляє також особливі стану ,як наявність у всіх розрядах зберігається слова одиниць або нулів.

2. КОДИ КОРЕКЦІЇ МОДУЛЬНИХ ПОМИЛОК

2.1 Основні терміни та визначення

Зберігання інформації в напівпровідникових ЗП осуществляетсяв вигляді кодів. Будь-який код характеризується наступними параметрами.

Підстава q - число різних елементарних символів для побудови коду.

Значність n - число елементів, що утворюють кодову комбінацію. Далі розглянемо рівномірні коди, тобто коди, все комбінації яких мають однакове число символів.

Потужність M - число різних кодових комбінацій при заданих q і n одно $M_{\max}=q^n$. Код, що містить всі ці комбінації, називається повним. При передачі або зберіганні повідомлень повним кодом число повідомлень дорівнює потужності коду. Якщо число кодових слів менше $M_{\max}$, тоді код називається надлишковим. Надмірне кодування дозволяє виявляти і виправляти помилки виникли при передачі або зберіганні повідомлень. Такі коди відомі як коригувальні.

Для побудови коригуючого коду з повного коду вибирають *M* комбінацій, віддалених один від одного на якомога більшу відстань. Ступінь віддаленості будь-яких двох кодових комбінацій оцінюється відстанню Хеммінга *d*, що дорівнює кількості незбіжних позицій. Вибрані комбінації вважаються дозволеними, решта - забороненими. При відмовах і збоях елементів пам'яті кодова комбінація переходить в заборонену. Остання повинна ототожнюватися споживачем з найближчої дозволеної комбінацією. Тому для виявлення *t* помилок рассояніе Хемминга має бути не менше $d = t + 1$, а для виправлення не менше $d = 2t + 1$. Платою за можливість виправлення помилок є зниження швидкості передачі або числа збережених повідомлень. Кількісно це оцінюється відносною надмірністю коду, що дорівнює $R=(n-k)/n$, де $k=]\log_q M[$,а $]x[$ - найближче ціле, більше чи менше x . Величина *k* - число інформаційних символів коду , а $r=n-k$ - число перевірочних символів. Серед коригувальних кодів найбільшого поширення набули лінійні (групові) коди з символами з

деякого кінцевого поля *GF(q)*. Лінійний *(n,k)* - код визначається як підпростір розмірністю *k* лінійного *n*-мірного простору над *GF(q)*. Код можна задати перерахуванням всіх кодових (дозволених) векторів або тільки перерахуванням базисних векторів підпростору. Другий спосіб набагато компактніше і тому отримав найбільше застосування. Сукупність базисних векторів будемо далі записувати у вигляді матриці *G* розмірністю *k* x *n* з одиничною подматріцей в перших *k* рядках і стовпцях .

$$G = [\, I \mid P \,] \qquad (2.1)$$

Матриця *G* називається породжує матрицею лінійного коригуючого коду в наведеному - ступінчастою формі. Кодові слова є лінійними комбінаціями рядків матриці *G* (крім слова, що складається з нульових символів) . Кодування полягає в множенні вектора повідомлення довжини до на породжує матрицю за правилами матричного множення, коли всі операції виконуються за модулем два. Очевидно, що при цьому перші до символи кодового слова дорівнюють відповідним символам повідомлення, а останні *r* символів утворюються як лінійні комбінації перших. Для будь-якої породжує матриці *G* існує матриця *H* розмірності *r***n*, що задає базис нульового простору коду і задовольняє рівності

$$GH^T = O \qquad (2.2)$$

Матриця *H* називається перевірочної. Вона дорівнює

$$H = [\, -P^T \mid I \,] \qquad (2.3)$$

В останньому виразі *I* - одинична матриця порядку *r*. Результат множення зчитує з пам'яті вектора - повідомлення на транспоновану перевірочну матрицю називається синдромом (вектором помилки) *S*. Якщо синдром дорівнює нулю *S=0*, то вважається в силу (2.2), що помилок в зчитуваному повідомленні немає. При ненулевом синдромі з його вигляду можна виявити або виправити помилку. Коди, задаючи-ються матрицями *G* і *H* у вигляді (2.1) і (2.2) є систематичними, або роздільними, оскільки в них відоме місце розташування інформаційних і перевірочних розрядів. Нульовий простір коду з породжуючої матрицею *G*, в свою чергу, також утворює код. Його базисом служать рядки матриці *H*. Цей код - двоїстий коду з породжує матрицею *G*.

Великим класом лінійних кодів є циклічні коди. Код називається циклічним, якщо разом з кожним кодовим словом він містить і всі його циклічні перестановки. Циклічні коди зазвичай задаються за допомогою генераторного і перевірочного поліномів, на підставі яких легко знаходяться генераторна і перевірочна матриці.

2.2 Модульні коди

Ці коди призначені для корекції помилок в пристроях з модульною структурою, коли пам'ять будується на приладах з многоразрядной (b-розрядної) організацією. При цьому всі повідомлення довжини n ділиться на N груп символів (модулів) з b символів в кожній групі.

Вихід з ладу одного приладу пам'яті пристрою спотворює лише подблок (модуль) кодового слова, межі якого відомі на відміну від помилок. Тому часто модулі помилок називають фазованими пакетами помилок.

2.2.1 Виявлення одиночних модулів помилок

Коди для виявлення одиночних модульних помилок легко побудувати на основі кодів з простою перевіркою на парність, об'єднавши i-е символи кожного модуля в кодове слово коду з відстанню d = 2. Перевірочна матриця такого коду має розмірність bxn і складається з поодиноких матриць I_b порядку b:

$$H=[\ I_b \mid I_b \mid \text{---} \mid I_b\ ].$$

2.2.2 Виправлення одиночних модульних помилок

Для виправлення одиночних модульних помилок довжиною *b* код повинен мати не менше *2b* перевірочних символів. Найбільший інтерес представляють коди з мінімальною надмірністю (*2b*) і мінімальною щільністю одиниць в перевірочної матриці. Останнє дозволяє досягти максимальної швидкодії при мінімальній складності схем кодування і декодування. Далі розглянуті тільки такі коди. Їх конструкції даються теоремами 2.1, 2.2.

Позначимо через M двійкову (2b + 1) xb матрицю виду

$$\mathbf{M} = \begin{bmatrix} 0\,0\,\ldots\,0 \\ \text{------} \\ \mathbf{I}_b \\ \text{------} \\ \mathbf{I}_b \end{bmatrix}$$

де Ib - одинична матриця порядку b; Ib \ - матриця, отримана з Ib зміною порядку проходження рядків на зворотний. Нехай Mji - матриця, отримана з M циклічним зрушенням j верхніх рядків на і позицій вниз і відкиданням останнього рядка.

Теорема 2.1. Матриця виду вида $\mathbf{H}=[\,\mathbf{M}_p^{\,0} \mid \mathbf{M}_p^{\,'} \mid \ldots \mid \mathbf{M}_p^{\,N-1}]$, де p - будь-яке просте число $b+1 \le p \le 2b+1$, а $N \le p$ задається код з виправленням одиночних модулів помилок і мінімальною щільністю перевірок.

Нехай тепер

$$\mathbf{L}^{'} = \begin{bmatrix} 0\,0\ldots 0 \\ \text{------} \\ \mathbf{I}_b \end{bmatrix},$$

а L_I - матриця, отримана з $L^{'}$ циклічним зрушенням всіх її $b + 1$ рядків на і рядків вниз і відкиданням останнього рядка.

Теорема 2.12. Матриця виду

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_b \mid \mathbf{I}_b \mid \mathbf{I}_b \mid \ldots \mid \mathbf{I}_b \mid \mathbf{I}_b \mid 0 \\ \text{----------} \\ \mathbf{L}_0 \mid \mathbf{L}_1 \mid \mathbf{L}_2 \mid \ldots \mid \mathbf{L}_{N-3} \mid \mathbf{I}_b \mid \mathbf{I}_b \end{bmatrix}$$

де $b+1 = p$, $N \le b+2$ задає код з виправленням одиночних модулів помилок, мінімальною щільністю перевірок і попарной залежністю контрольних символів

2.2.3 Виправлення багаторазових модульних помилок

Для виправлення багаторазових модульних помилок можна використовувати коди над полем GF (q) (q - просте число), що виправляють t і менше помилок. Якщо береться код над полем GF (2^b). Те кожен символ цього коду $a_i \in GF\ (2^b)$ замінюється його поданням у вигляді вектора b-мірного простору над полем *GF* (2). Якщо береться код над полем *GF* (*q*), то кожен символ коду замінюється його двійковим b-розрядних поданням (b=] $\log_q(p\text{-}1)$ [ ).

2.3. Алгоритми виправлення незалежних помилок і блоки корекції

Схема корекції помилок за допомогою коригувальних кодів показана на рис..2.1. Повідомлення (інформаційні символи), призначені для зберігання, надходять в кодер, перетворюються там в одне зі слів коду і записуються в ЗП. Лічені з ЗП символи надходять на входи декодера, який здійснює виправлення помилок у доглянутому слові і видає споживачеві скориговані інформаційні символи. Далі обмежимося розглядом тільки паралельних методів кодування і декодування, коли всі позиції вхідного вектора або кодового слова обробляються одночасно. Методи послідовного (посимвольного) декодування вимагають значних витрат часу і, як правило, неприйнятні для напівпровідникової пам'яті.

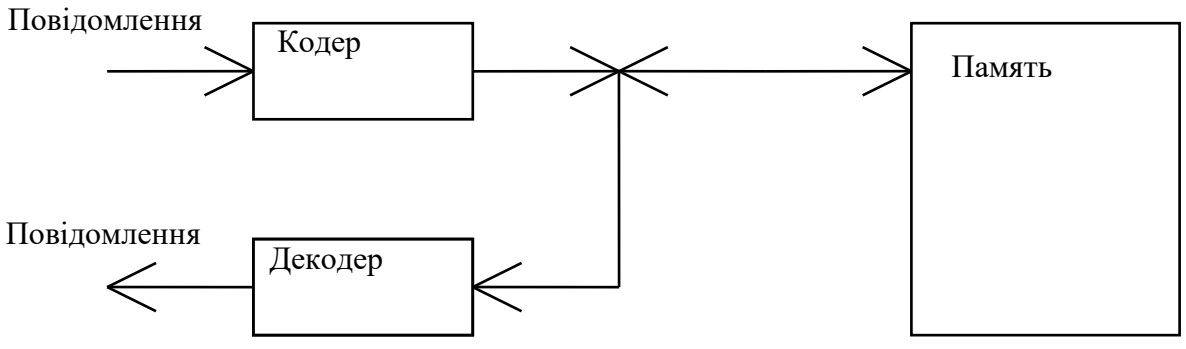

Рис.2.1. Структурна схема коррекції помилок

2.3.1 Кодування

Ьлок кодування (БК) лінійних кодів реалізує операцію множення вектора повідомлення А на породжуючу матрицю G, тобто AG = B. Оскільки найчастіше в якості матриці G використовується матриця в наведеному-ступеневу вигляді G=[I | G'], то при множенні А на I не відбувається перетворення А, а при множенні А на G' обчислюються символи перевірочних розрядів A_r. Таким чином, в пам'ять заноситься слово B=[A | A_r], на перших *k*-позиціях якого розташовані вихідні дані, а на що залишилися *r*-позиціях - обчислені контрольні розряди..

При цьому *i*-й перевірки символ дорівнює сумі інформаційних символів повідомлення, що знаходяться на позиціях ненульових елементів *i*-го стовпця матриці G'. Наприклад, для подовженого коду Хеммінга (8; 4) з породжуючою матрицею:

$$G = \begin{bmatrix} 1\,0\,0\,0 & 1\,1\,0\,1 \\ 0\,1\,0\,0 & 0\,1\,1\,1 \\ 0\,0\,1\,0 & 1\,0\,1\,1 \\ 0\,0\,0\,1 & 1\,1\,1\,0 \end{bmatrix} \qquad (2.4)$$

Контрольні символи Ar = (a5, a6, a7, a8) обчислюються виходячи з інформаційних символів A = (a1, a2, a3, a4) наступним чином:

$$a_5 = a_1 + a_3 + a_4; \qquad a_7 = a_2 + a_3 + a_4; \qquad (2.5)$$

$$a_6 = a_1 + a_2 + a_4; \qquad a_8 = a_1 + a_2 + a_3;$$

На рис. 2.2 наведено кодуючий блок коду (8; 4), який реалізує систему рівнянь (2.5). при обчисленні шостого і восьмого символів використовується одна і та ж сума a1 + a2. Її досить обчислити один раз. У ряді випадків облік таких повторень може дати істотну економію обладнання, однак призводить до розмноження помилок при відмові суматора.

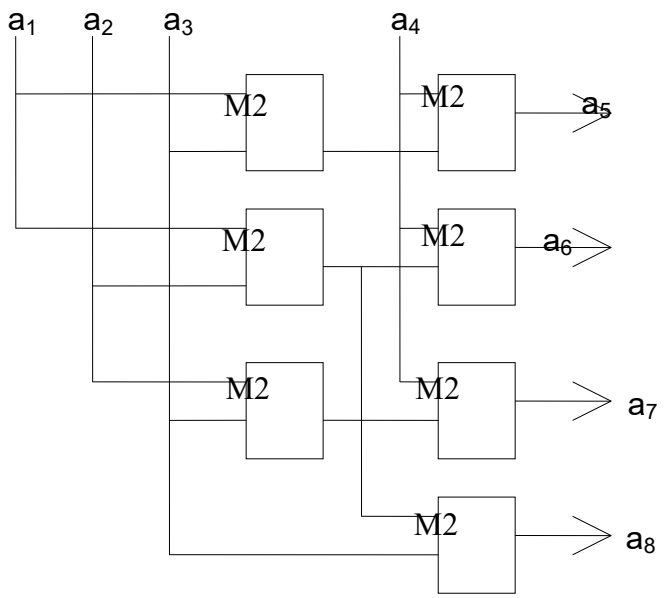

Рис. 2.2. Кодуючий блок коду Хемінга(8; 4)

2.3.2 Декодування

Структура декодируючого блоку значно складніше, ніж кодуючого. Процедура декодування може реалізуватися як апаратними, так і програмними засобами за допомогою мікро-ЕОМ. У першому випадку намагаються мінімізувати загальний обсяг обладнання, у другому - кількість необхідних для декодування обчислювальних операцій.

Найбільш важливими методами паралельного декодування є наступні: декодування по мінімуму відстані, декодування по синдрому, мажоритарну декодування.

2.3.2.1 Декодирование по мінімуму відстані

При цьому методі декодування лічений з пам'яті вектор порівнюється з усіма словами використовуваного коригуючого коду і обчислюється відстань між ними. Лічені вектор ототожнюється з тим кодовим словом, від якого він був знищений на мінімальну відстань.

При незалежних і симетричних помилках цей спосіб дозволяє мінімізувати загальну ймовірність отримання спотвореного повідомлення, однак складний в реалізації. Дійсно, операція порівняння двох *n*-розрядних слів здійснюється або шляхом посимвольного підсумовування цих слів по модулю два і знаходження всієї суми (якщо використовується алфавіт 0, 1), або шляхом обчислення їх скалярного твори (якщо використовується алфавіт 1, 1). Це вимагає не менше *n* арифметичних операцій додавання (віднімання) двох чисел. Загальна ж кількість арифметичних операцій при програмної реалізації буде пропорційно потужності коду і оцінюється величиною $n\ 2^k$. Аналогічне значення має і апаратурна складність, якщо лічений повідомлення порівнюється одночасно з усіма кодовими словами. Наведені підрахунки показують, що декодування по мінімуму відстані доцільно використовувати лише в кодах з малим числом інформаційних символів.

Описаний алгоритм є найбільш загальним і не враховує алгебраїчних і комбінаторних особливостей коду. У той же час майже очевидно, що складність декодування можна зменшити за рахунок багаторазового використання результатів проміжних обчислень (або розгалуження схеми).

2.3.2.2 Декодування по синдрому

Процедура декодування по синдрому складається з трьох етапів: обчислення синдрому $S= AH^T= (S_1, S_2, \ldots, S_r)$, визначення по синдрому вектора помилки, виправлення повідомлення шляхом підсумовування його з вектором помилки.

Можна показати, що при незалежних і симетричних помилках декодування по синдрому еквівалентно з точки зору достовірності декодування по мінімуму відстані. Перший етап реалізується блоком обчислення синдрому

(БОС), який може доповняться блоком аналізу синдрому (БАС). Сигнали, що надходять цього блоку з'являються, якщо відбулися невиправні помилки і просто помилка.

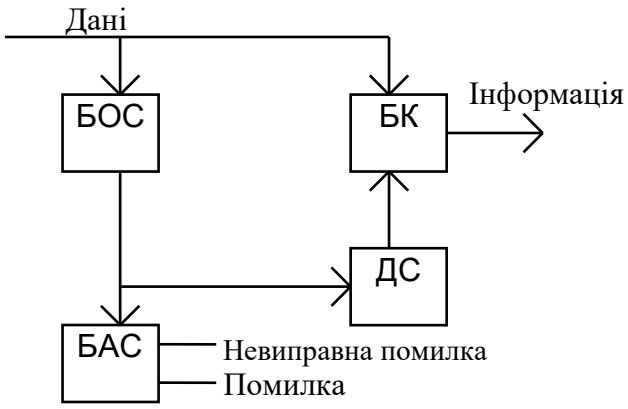

Рис.2.3. Декодирование по синдрому

Другий етап, як правило, реалізується дешифратором синдрому (ДС), який кожному значенню синдрому ставить у відповідність сукупність вихідних сигналів, відповідних вектору-помилку. Сигнали з виходу дешифратора можуть бути також використані для визначення місця розташування помилки при ремонті або заміні який відмовив блоку. Це дозволяє постійно підтримувати високу надійність системи, тому що своєчасне усунення відмови робить кожен наступний відмова "першим". У ряді випадків доцільно також введення додаткового блоку пам'яті для зберігання синдромів. Це дає можливість отримати картину розподілу помилок по полю основного накопичувача, необхідну для визначення виду помилки (відмова або збій) і ремонту відмовившого блоку. Нарешті, блок корекції (БК) являє собою набір суматори за модулем два, на яких відбувається складання вектора помилки зі лічених словом. Як сумматоров можуть бути використані тригери вихідного регістра ЗП. У цьому випадку корекція здійснюється подачею одиничного сигналу з виходу дешифратора на рахунковий вхід тригера.

Найбільш складним є блок обчислення синдрому і дешифратор, а найбільш трудомістким другий етап декодування, в якому по синдрому знаходиться вектор-помилка. Складність цієї процедури для довільного коду оцінюється величиною $n2^r$. Тому декодування по синдрому доцільно в кодах з невеликим числом перевірочних символів.

Розглянемо особливості реалізації цих блоків. Обчислення синдрому полягає в множенні ліченого вектора на транспоновану перевірочну матрицю H^T. Тому для спрощення блоку обчислення синдрому слід зменшити число одиниць в перевірочної матриці. Це дозволяє скоротити число суматорів або число елементарних складань при роздільному обчисленні кожного символу синдрому. Оскільки рядки перевірочної матриці утворюють базис нульового простору коду, то задача зводиться до перебору всіх можливих базисів. Громіздкість цього завдання очевидна, тому на практиці повний перебір практично не використовується, а задовольняються приватними рішеннями, отриманими методом проб і помилок.

Наступний крок полягає в використанні результатів обчислення одного символу синдрому для отримання іншого символу. Ситуація тут аналогічна розглянутої раніше.

При практичній реалізації блок кодування доцільно поєднувати з блоком обчислення синдрому, оскільки в більшості ЗП операції запису і зчитування рознесені в часі. При поєднанні в якості кодера фактично використовується БОС. При цьому на входи, відповідні перевірочним розрядам, подаються нульові сигнали, а перевірочні символи коду знімаються з виходів синдрому. Як зазначалося раніше, використання мінімізованої схеми призводить до розмноження помилок при відмові елемента з ветвящимся виходом.

Складність схеми реалізації дешифратора залежить від кратності виправляє помилок. Істотне спрощення досягається відмовою від дешифрування синдромів помилок в перевірочних розрядів. У більшості випадків це допустимо, тому що досить виправити помилки тільки в

інформаційних розрядах. Незважаючи на це складність реалізації другого етапу декодування залишається високою.

2.3.2.3 Мажоритарне декодування

Цей метод декодування відносно простий як в описі, так і в реалізації, однак не завжди реалізує всі коригуючі властивості коду. Ідея мажоритарного декодування лінійного коду базується на системі перевірочних рівностей. А саме, в кодах з мажоритарних декодуванням кожен символ може бути виражений через інші символи декількома способами. Це дозволяє для визначення справжнього значення символу скористатися принципом більшості (мажоритарних принципом).

Складність мажоритарного декодування залежить від виду перевірочних рівнянь, їх числа і кількості символів, що входять в кожне рівняння. Найбільш доцільним є випадок, коли перевірочні рівняння містять в правій частині тільки два члена і спотворення будь-якого з символів правій частині призводить до спотворення тільки одного рівняння. В цьому випадку для виправлення t помилок досить мати $2t+1$ незалежних перевірочних рівнянь.

При великій кратності виправляє помилок істотним є питання про реалізацію мажоритарного елементу. У схемах з алфавітом (0, 1) мінімальну асимптотическую складність має мажоритарний елемент, який представляє собою комбінацію паралельного лічильника і дешифратора.

3. HDL-МОДЕЛЬ ПРИСТРОЮ

3.1 Корекція модульних помилок

В блоках пам’яті ВІС ЗП з однорозрядною організацією N*1, де N – число слів в матриці ЗП, основні категорії помилок виникають по відношенню до збоїв елементів пам’яті, і до помилок відмов. Тому використання пристроїв корекції поодиноких і двійкових помилок в блоках пам’яті однорозрядних ЗП, необхідна міра для ефективного збереження інформації.

В блоках пам’яті ВІС ЗП з багаторозрядною організацією ситуація дещо інша. Використання пристроїв корекції на основі коду Гемінга для систем з багаторозрядною організацією є некоректним, через суттєві помилки в виправленні помилок збереження даних в наслідок частих відмов цілих пристроїв.

Такі помилки є найбільш ймовірними серед постійних однобітних помилок, помилок стовбця і рядка, помилок всього пристрою. Тому при виході із строю хоча б одного пристрою в блоку пам’яті можуть бути помилковими зразу декілька розрядів слова. Такі помилки називають модульними помилками.

Ефективним засобом боротьби з модульними помилками є застосування кодів для корекції модульних помилок. Модульні коди дозволяють не тільки виявляти і визначати місцеположення відмовив модуля, а й виробляти виправлення помилок, як з-за спотворення модуля кодового слова, так і окремих одиночних помилок.

Пристрої корекції модульних кодів можуть бути побудовані на основі регістрів зсуву з зворотними зв'язками. Для високошвидкісних систем застосування таких пристроїв корекції неприпустимо через низьку швидкодію, обумовленого послідовним алгоритмом обробки. З точки зору збільшення швидкодії бажано застосування модульних кодів з мінімальною щільністю перевірок і мінімально можливим числом перевірочних розрядів.

Для виявлення одиночних модульних помилок використовується код з перевірочної матрицею, складеною з N + 1 одиничних матриць порядку b.

Синдром цього роду дорівнює вектору помилок відмовив модуля, але в ньому немає інформації про номер модуля.

Тому для виправлення такого роду помилок необхідно виконати такі дії при появі такої помилки:

1. Процесор інвертує пряме слово;
2. Знову його записує по тій же адресі;
3. Зчитує інвертоване слово і порівнює його з прямим, яке збереглося в додатковому регістрі.

Оскільки відмовили символи однакові як для прямого, так і інвертованого слова, то відбувається виявлення місця розташування відмовив модуля і виправлення прочитаної інформації шляхом підсумовування її з обчисленим синдромом. Платою за просту реалізацію є низька швидкодія в режимі виправлення інформації.

Збільшення швидкодії досягається застосуванням кодів для виправлення модулів помилок. Розглянемо реалізацію цих кодів на прикладі коду довжини n = 24 з чотирма модулями розрядності b = 4 (дані параметри найбільшим чином задовольняють потребам практики).

Код можна використовувати як для виявлення, так і для корекції помилок. Так, для виявлення модуля помилок довжиною b = 4 досить використувати перші чотири контрольних розрядів. Чотири розряди служать для виправлення помилок. Контрольні символи цих розрядів залежать від контрольних символів перших чотирьох. Кодуюча схема показана на Рис. 3.1.

Декодування може виконуватися звичайним методом шляхом обчислення синдрому і його дешифрування. Необхідно проте врахувати, що розшифрувати слід всілякі комбінації помилок в межах модуля, що призводить до дуже великої складності дешифратора.

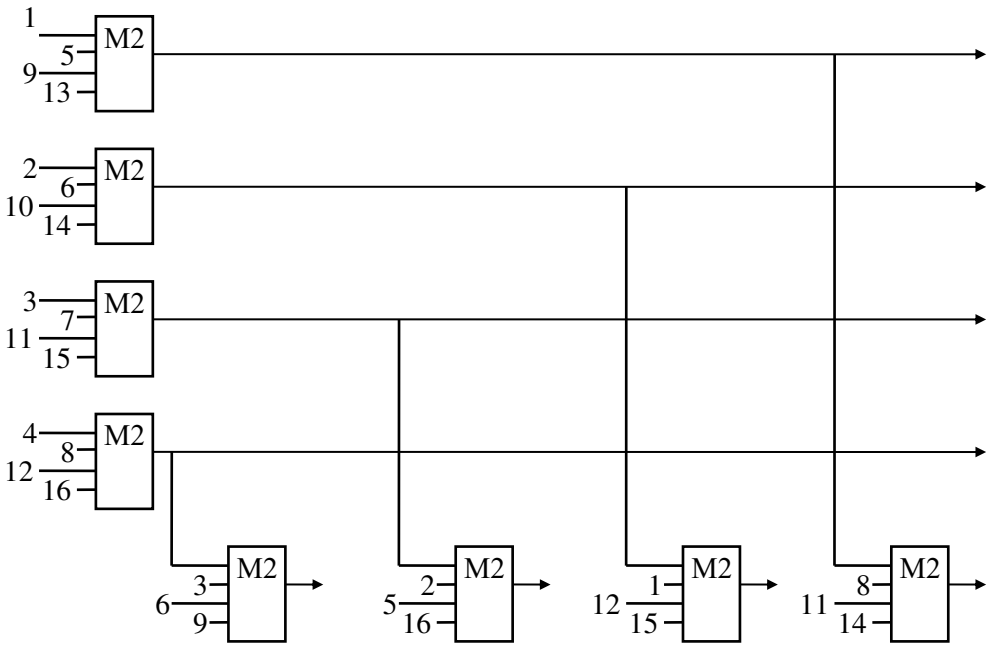

Рис. 3. 1. Кодирующая схема модульного коду

У той же час аналіз контрольної матриці показує, що конфігурація розташування помилок в модулі визначається першими чотирма розрядами синдрому, а номер модуля - залишившихся чотирма розрядами. Позначимо їх відповідно S1 і S2, тобто S = (S1, S2). Одна і та ж комбінація помилок, що вражає різні модулі, дає синдроми S2, що відрізняються тільки циклічним зрушенням, внесеним матрицею Ai. Зазначене властивість дозволяє побудувати наступний простий алгоритм декодування.

1. Обчислюється синдром S=(S1,S2).
2. Якщо S1=S2=0,то помилок немає.
3. Якщо S1=0 и S2<>0, то відмовив шостий модуль. Вектор помилки модуля дорівнює вектору S2.
4. Якщо S1=S2<>0,то відмовив пятий модуль. Вектор помилки модуля дорівнює вектору S1.
5. Якщо S1<>S2 тоді не один із векторів рівен нулю, тоді обчислюються окремі синдроми: Si=S1Ai, де i=1,2,3,4.
6. Окремі синдроми порівнюються із значенням S2, і визначається i, для якого виконується рівність Si = S2. Знайдене значення i, і є номер відмовившого модуля. Вектор помилки модуля дорівнює S1.

7. Проводиться корекція результату шляхом підсумовування зчитаного з і-го модуля сигналу з вектором S1.

На рис.3.2 наведено структурну схему декодуючого блоку, що реалізує даний алгоритм декодування.

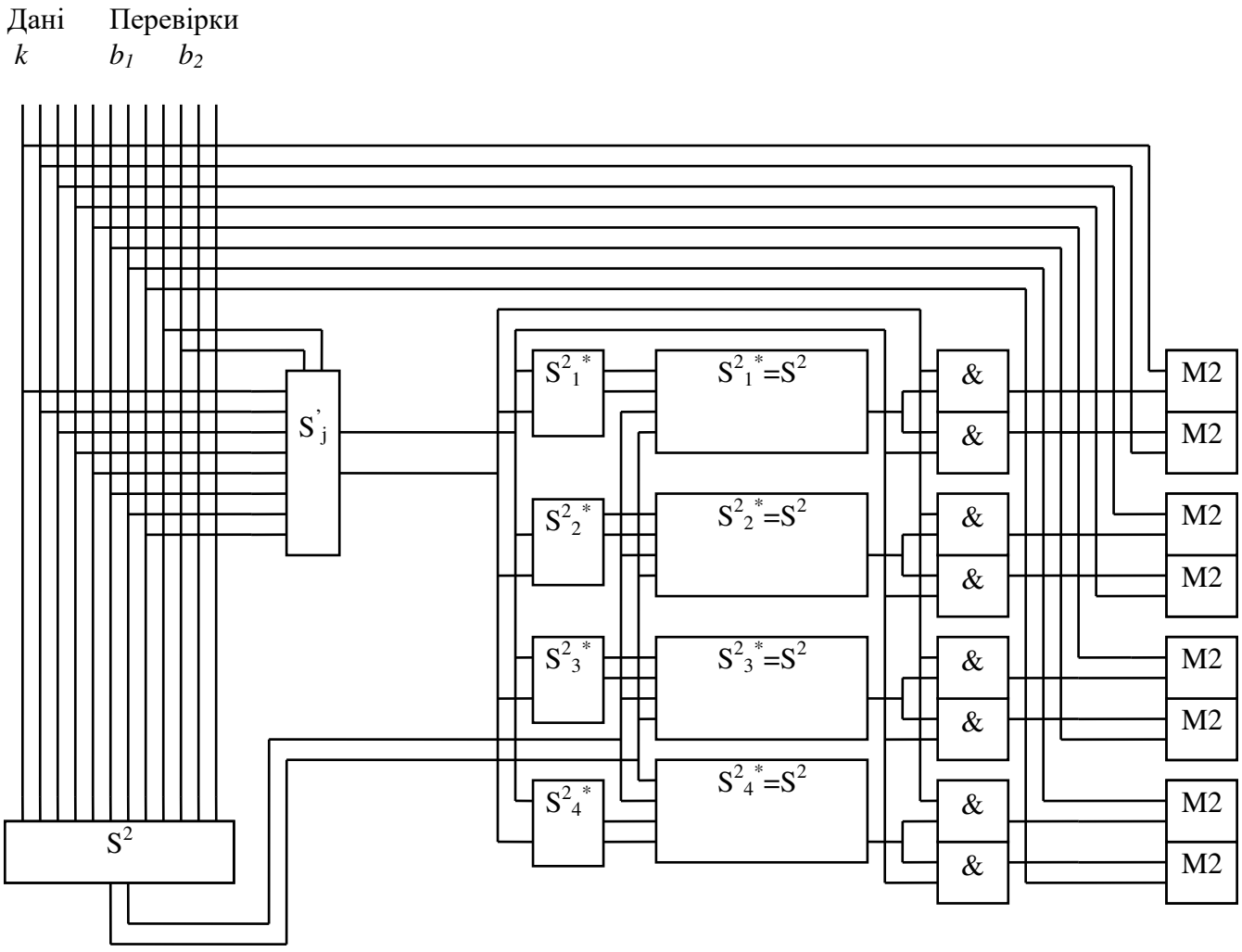

Рис. 3.2. Структурна схема декодирующего блоку для корекції одиночних модульних помилок.

Враховуючи, що повний збій пристрою пам'яті призведе до появи лише одного (нульового) сигналу на кожному виході, складність схем кодування та декодування може бути зменшена. У цьому випадку для побудови процедури фіксації єдиного модуля помилок можуть бути використані такі факти:

1. Синдром коду для виявлення одного модуля дорівнює вектору помилок модуля, що виходить з ладу.

2. Несправний модуль можна ідентифікувати за його вихідним сигналом [11 ... 1].

Слід мати на увазі, що справні модулі також можуть зберігати інформацію [11 ... 1]. Такі модулі повинні бути обладнані спеціальною «міткою».

Зважаючи на ці міркування, блок пам'яті повинен складатися з модуля N + 1 довжиною слова b1, який працює з кодом виявлення помилок, і модуля мітки довжиною слова b1 = N (друга перевірка) з довжиною слова b1 = N, в якому зберігатиметься інформація про наявність поодиноких слів в інформаційних N модулях. Якщо одне слово зберігається в і-му інформаційному модулі, то в і-му місці відповідного слова в модуль міток заноситься одиниця.

Процедура декодування відбувається наступним чином:

1. Якщо синдром S = 0, то вважається, що помилок немає або вони відбулися в другому перевірочному модулі. Ніяких дій не проводиться, дані з інформаційних модулів надходять споживачеві;

2. Якщо синдром S <> 0, то вихідні сигнали кожного модуля аналізуються на відповідність [11 ... 1]. При цьому можливі два результати:

А) ніяких дій не проводиться і інформація видається споживачеві, якщо вихід першого перевірочного модуля дорівнює [11 ... 1], вихід інформаційного модуля дорівнює [11 ... 1], і в відповідному розряді другого перевірочного модуля записана одиниця;

Б) проводиться корекція шляхом підсумовування синдрому з вихідним вектором модуля, якщо вихідний сигнал дорівнює [11 ... 1] і в відповідному розряді перевірочного модуля записаний нуль.

Схема, що реалізує цей алгоритм, наведена на рис. 3.3. Блок аналізу визначає, на вході яких модулів є поодинокі сигнали і чи відповідають вони його вміст в них інформації шляхом порівняння з зчитується інформацією відповідного розряду другого перевірочного модуля. При розбіжності на одному з входів блоку аналізу встановлюється сигнал лог.1, який відкриває *b*

відповідних елементів *i* спотвореного модуля, і обчислений синдром виправляє на суматорах по модулю два зчитується з пам'яті інформацію в перекрученому

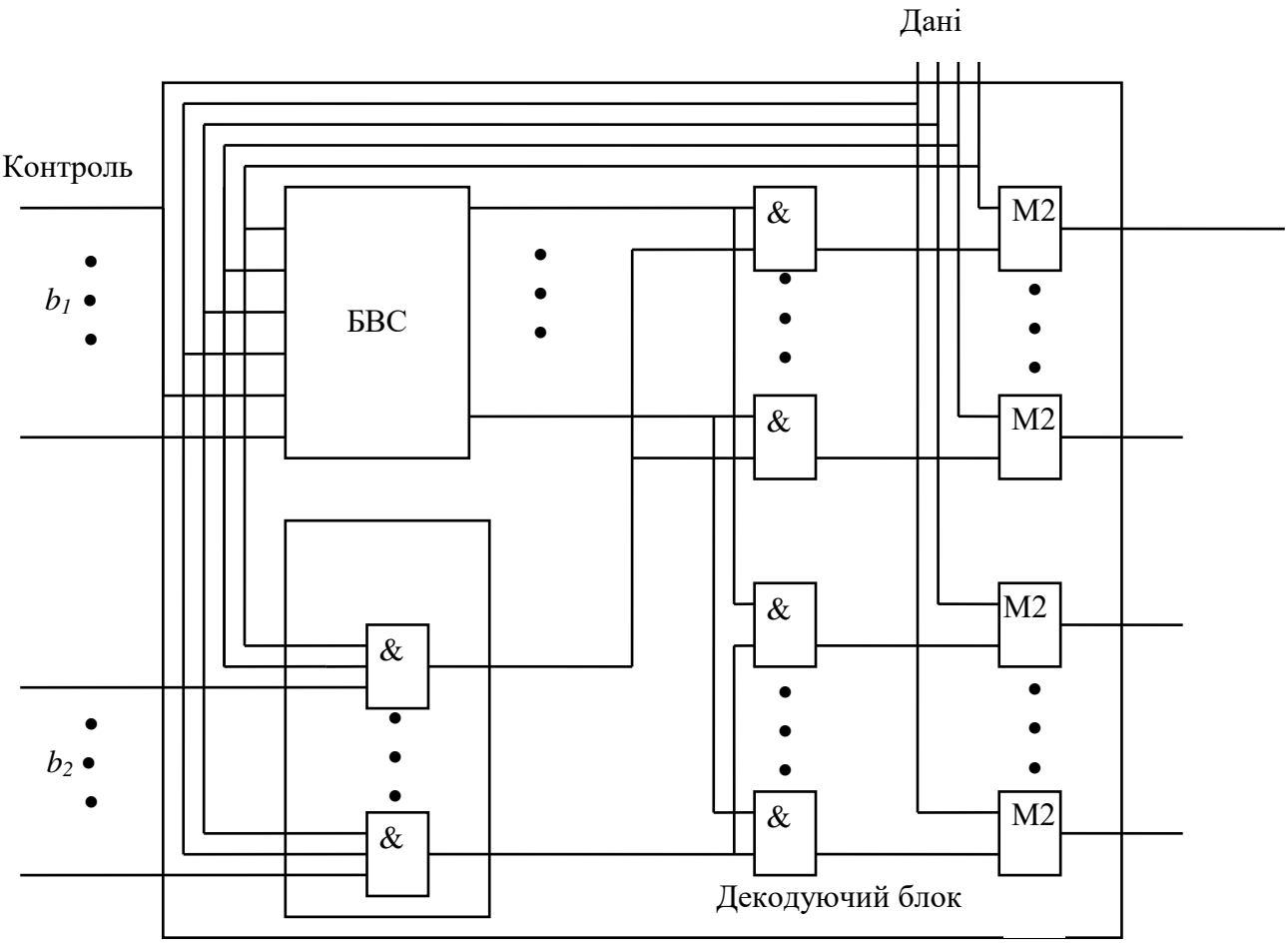

Рис. 3.3 Структурна схема декодуючого блоку для корекції одиничних модулів помилок

модулі. При наявності помилок в першому і другому перевірочних модулях на виходах елементів *i* завжди присутні сигнали лог.0 і виправлення зчитувальних даних на суматорах по модулю два не проводиться.

3.2 Блоки HDL-моделі

Блок кодування коригуючих кодом Хеммінга (БК)

Блок кодування коригуючих кодом Хеммінга виконує роль перетворювача інформаційних бітів в модульну послідовність, в якій кожен з

модулів дорівнює 4 бітам. В цій модульній послідовності до кожних 4 бітів вхідного слова додається 2 контрольних біта для перевірки при декодуванні.

Блок кодування коригуючих кодом Хеммінга використовує один вхід і вихід. На вхід подається сигнал до 16 біт. Після кодування на виході данні складаються з 24 біт. Також блок кодування використовує додатковий елемент Ctrl_bit, який вираховує контрольні біти з входу. Навхід цього контролюючого елементу подається 16 бітів даних. Після обробки цих даних на вихід він подає 8 контрольних бітів, які додаються до 16 бітів даних пройдених через кодер і на виході ми отримуємо 24 біта. (Код цього блоку можна знайти в ДОДАТКУ Б на сторінці 1)

Блок декодування (БД)

Блок декодування необхідний для того щоб модульну послідовність з запамятовульного пристрою перетворити в інформаційну послідовність з виявленням одиночних модульних помилок і подвійних модульних помилок. Ця модульна послідовність складається з 4 інформаційних бітів і 2 контролюючих бітів. Аналізуючи 2 контрольних біта декодер на основі цієї інформації виявляє одиночні модульні помилки і подвійних модульні помилок. Після цього декодер виправляє виявлені помилки і на виході декодера отримуємо правельну інформаційну послідовність.

Блок декодування використовує один вхід і два виходи. На вхід декодуючого блоку подається модульна послідовні, яка містить 24 розряди із яких 16 інформаційних розрядів і 8 контрольних розрядів.Сам блок декодування складається з трьох основних елементів: контрольного блоку, блоку обчислення синдрому, блоку корекції інформаційних бітів. Про кожен із цих блоків поговоримо далі окремо.

Модульна послідовність отримана із запам’ятовуючого пристрою на вході декодуючого пристрою, розділяється на 16 інформаційних бітів і 8 контрольних бітів. Інформаційні біти із входу декодера переміщаються до блоку корекції і до блоку обчислення контрольних розрядів, а 8 контрольних бітів переміщаються до блоку обчислення контрольних розрядів. Після цього

контрольні біти передаються до блоку обчислення синдрому, а вже після цього разом із інформаційними бітами, контрольні біти разом з інформацією отриманою на попередніх блоках передаються до блоку корекції помилок. Із блоку корекції помилок виходять два сигнали. Один із сигналів є правильною інформаційною послідовністю,який складається із 16 інформаційних бітів. Другий сигнал повідомляє іде до контролюючого пристрою і може повідомити про виявлення помилки. Цими двома виходами і закінчується декодуючий пристрій. (Код цього блоку можна знайти в ДОДАТКУ Б на сторінці 2-3)

Блок обчислення контрольних розрядів (БОКР)

Блок обчислення контрольних розрядів складається з одного входу і одного виходу. На вхід блоку обчислення контролюючих розрядів подається інформаційний потік який складається із 16 бітів інформації. Після цього 16 інформаційних бітів розділяються на модулі, які складаються з 4 бітів. Кожен з цих модулів передається на блок множення m-розрядних двійкових слів по модулю многочлена x^4+x+1, в якому і виконується робота по обчисленюю контрольних розрядів. На кожний модуль створюється по 2 контрольних розряди. На виході блоку обчислення контрольних розрядів відправляється 8 контролюючих розрядів. (Код цього блоку можна знайти в ДОДАТКУ Б на сторінці 7)

Блок множення m-розрядних двійкових слів по модулю многочлена x^4+x+1 (MUL)

Блок множення m-розрядних двійкових слів по модулю многочлена x^4+x+1 виконує обчислення контрольних розрядівдля кожного модулю. На вхід цього блоку подається 2 сигнали. Перший вхід оповіщає як потрібно рахувати той чи інший модуль. Самі модулі задаються другим входом, який саме це модуль, яке значення кожного з бітів з яких він складається. Внаслідок того що модуль складається із 4 бітів на вхід модуля використовується 4 біта. (Код цього блоку можна знайти в ДОДАТКУ Б на сторінці 8)

Блок обчислення синдрому (БОС)

Блок обчислення синдрому використовується для обчислення векторів синдрому. Цей блок складається із двух входів і двух виходів. На перший вхід відбувається передача 8 контрольних розрядів, які були записані в запам’ятовуючий пристрій разом із інформаційним бітами в модулях. На другий вхід передається 8 контрольних розрядів, які отримуються після повторного обчисленння в блоці обчислення контрольних розрядів в декодері. На першому виході відбувається порівняння перших 4 бітів контрольних розрядів, які отримали при кодуванні в кодуючому пристрої з першими 4 бітами, які були отримані в блоці обчислення контрольних розрядів в декодуючому пристрої. На другому виході відбувається порівняння останніх 4 бітів контрольних розрядів, які отримали при кодуванні в кодуючому пристрої з останніми 4 бітами, які були отримані в блоці обчислення контрольних розрядів в декодуючому пристрої. (Код цього блоку можна знайти в ДОДАТКУ Б на сторінці 3)

Блок множення першої половини синдрому на кожен з стовпців другого рядка контрольної матриці (ext_syndrome)

Блок множення першої половини синдрому на кожен з стовпців другого рядка контрольної матриці складається з одного входу і чотирьох виходів. Вхід блоку складається з 4 бітів. На них подається 4 байти першої половини синдрому. Кожен з байтів синдрому, які були отримані, перемножаються за допомогою блоку множення m-розрядних двійкових слів по модулю многочлена x^4+x+1. Після цього отримані результати відправляються на 4 виходи кожен з яких складається з 4 бітів. (Код цього блоку можна знайти в ДОДАТКУ Б на сторінці 4)

Блок визначення помилки (БВП)

Блок визначення помилки повертає номер модуля в якому помилка. Він складається з двох входів і одного виходу. На перший вхід подається, який складається з 4 бітів, результат порівняння перших 4 бітів контрольних розрядів, які отримали при кодуванні в кодуючому пристрої з першими 4 бітами, які були отримані в блоці обчислення контрольних розрядів в

декодуючому пристрої. На другий вхід подається, який складається з 4 бітів, порівняння останніх 4 бітів контрольних розрядів, які отримали при кодуванні в кодуючому пристрої з останніми 4 бітами, які були отримані в блоці обчислення контрольних розрядів в декодуючому пристрої. Ці результати були отримані із блоку обчислення синдрому і називаються синдромом. Конфігурація розташування помилок в модулі визначається першими чотирма розрядами синдрому, а номер модуля - залишившихся чотирма розрядами. Позначимо їх відповідно S1 і S2.

Перші 4 біти, які ми отримали на першому вході блоку відправляються до блоку множення першої половини синдрому на кожен з стовпців другого рядка контрольної матриці (ext_syndrome). Після цього відбувається порівняння синдрому, який складаєтьсяз перших 4 розрядів синдрому, те що ми отримали від блоку множення першої половини синдрому на кожен з стовпців другого рядка контрольної матриці, і останніх чотири розряди синдрому. Порівняння відбувається наступним чином:

1. Якщо S1=S2=0,то помилок немає.
2. Якщо S1=0 и S2<>0, то відмовив шостий модуль. Вектор помилки модуля дорівнює вектору S2.
3. Якщо S1=S2<>0,то відмовив пятий модуль. Вектор помилки модуля дорівнює вектору S1.
4. Якщо S1<>S2 тоді не один із векторів рівен нулю, тоді обчислюються окремі синдроми: Si, де i=1,2,3,4.
5. Окремі синдроми порівнюються із значенням S2, і визначається i, для якого виконується рівність Si = S2. Знайдене значення i, і є номер відмовившого модуля. Вектор помилки модуля дорівнює S1.

Після цього номер відмовившого модуля відправляється до виходу блоку. (Код цього блоку можна знайти в ДОДАТКУ Б на сторінці 5)

Блок корекції визначеної помилки (БКВП)

Блок корекції визначеної помилки останній блок декодера. Він складається з 3 входів і 2 виходів. На перші два входи, які складаються із 4 бітів кождний, подається перша і друга половина синдрому відповідно. На третій вхід, який складаєтьсяіз 16 бітів подається інформаційні біти, які були в ЗП.

На початку роботи блоку виконується виклик блоку comparsion, який визначає в якому модульному блоці помилка. Цей блок результатом роботи показує номер модуля в якому є помилка, або якщо повертає 0 то помилок немає.

Після цього проводится корекція результату шляхом підсумовування зчитаного з і-го модуля сигналу з вектором S1 таким чином:

1. Якщо помилка в першому модулі, тоді перші чотири біти модульної інформаційної послідовності підсумовуються з вектором S1, а інші 3 модулі не змінюються.
2. Якщо помилка в другому модулі, тоді другі чотири біти модульної інформаційної послідовності підсумовуються з вектором S1, а перші чотири біти і 2 наступних модулі не змінюються.
3. Якщо помилка в третьому модулі, тоді треті чотири біти модульної інформаційної послідовності підсумовуються з вектором S1, а перші 2 модулі і останній модуль не змінюються.
4. Якщо помилка в четвертому модулі, тоді останні чотири біти модульної інформаційної послідовності підсумовуються з вектором S1, а перші 3 модулі не змінюються.
5. Якщо помилок немає, то інформаційна послідовність залишається без змін.

Якщо блок comparsion результатом роботи повертає 7, тоді в модульній послідовності виникла подвійна модульна помилка.

(Код цього блоку можна знайти в ДОДАТКУ Б на сторінці 6)

3.3 Тестування HDL-моделі

При будь-якому результаті роботи на виході даного блоку і декодера в цілому інформаційна послідовність відправляється на перший вихід, який складається із 16 бітів, і далі на вихід із декодера. Якщо була отриманий сигнад про наявність подвійної помилки, тоді відправляється сигнал на другий вихід, який веде до контролюючого пристрою.

Використовуючи цю модель можна закодовувади та розкодовувати інформаційнаи потік, який сладається із 16 бітів, для того щоб зберігати дані на запам’ятовувальному пристрої, виправляти помилки, які виникають при записі і зчитуванні із запам’ятовувального пристрою.

В кінці розробки було перевірено модель на виконяння заданої задачі. Спочатку 16 бітний інформаційний текст було відправлено до блоку кодування

Delta	data_in	data_out	control_bits
0	(2D71)	(002D71)	(71)
1	(2D71)	(2D7171)	(71)
2	(2D71)	(2D7171)	(9D)
3	(2D71)	(2D719D)	(9D)

і він повернув результат, який показаний на рис. 3.4.

Рис.3.4 Результат кодуючого блоку

Згідно з цим інформаційний біти 2D71. які були використані для тестування, отримали додатково ще 8 бітів, які були обраховані в цьому блоці і стали рівними 9D, і в результаті була отримана модульна послідовність 2D719D. Таку послідовність можна записувати в запам’ятовуючий пристрій. Після цього при виведенні інформації із запам’ятовувального пристрою необхідно її декодувати і виправити в разі необхідності. Для цього модульну послідовність переводять до декодера. Якщо модульна послідовність не змінилася під час вводу і виводу інформації із запам’ятовуючого пристрою тоді

модульна послідовність не зміниться і передасться на вихід без 8 контрольних бітів. Такий результат із правильною комбінацією показаний на рис 3.5.

Delta	data_in	data_out	double_error	control_bits
0	(2D719D)	(UUUU)	U	(UU)
1	(2D719D)	(2D71)	0	(UU)
2	(2D719D)	(UD71)	0	(9D)
4	(2D719D)	(2D71)	0	(9D)

Рис 3.5 Результат роботи декодера із правильною вхідною комбінацією модульних бітів 2D71 і контрольних бітів 9D

Якщо модульна послідовність змінилася при процедурах запису і зчитування інформації тоді декодер буде змінювати вхідну модульну послідовність на правильну. Якщо помилка тільки в одному модулі тоді декодер її виправить і правильну модульну послідовність передасть на вихід без контрольних бітів. Такий результат із однією помилкою в одному модулі

Delta	data_in	data_out	double_error	control_bits
0	(2D319D)	(UUUU)	U	(UU)
1	(2D319D)	(2D31)	0	(UU)
2	(2D319D)	(UD31)	0	(DE)
4	(2D319D)	(6D31)	0	(DE)
5	(2D319D)	(2D31)	1	(DE)
6	(2D319D)	(2D71)	0	(DE)

показаний на рис 3.6.

Рис 3.6 Результат роботи декодера із однією помилкою в вхідної комбінації модульних бітів 2D31 і контрольних бітів 9D

У випадку коли помилок більше ніж одна, тоді декодер може повести себе різним чином. Якщо ці помилки тільки в одному модулі, тоді їх декодер виправить і правильну модульну послідовність передасть на вихід без контрольних бітів. Такий результат із декількома помилками в одному модулі показаний на рис 3.7.

Delta	data_in	data_out	double_error	control_bits
0	(28719D)	(UUUU)	U	(UU)
1	(28719D)	(2871)	0	(UU)
2	(28719D)	(U871)	0	(C7)
4	(28719D)	(7871)	0	(C7)
5	(28719D)	(2871)	1	(C7)
6	(28719D)	(2D71)	0	(C7)

Рис 3.7 Результат роботи декодера із двома помилками в одному модулі вхідної комбінації модульних бітів 2871 і контрольних бітів 9D

Якщо помилкок декілька і вони в різних модулях, тоді декодер не зможе виправити вхідну модульну комбінацію, але повідомить при наявність декількох модульних помилок. Такий результат із декількома помилками в одному модулі показаний на рис 3.8.

Delta	data_in	data_out	double_error	control_bits
0	(28619D)	(UUUU)	U	(UU)
1	(28619D)	(2861)	0	(UU)
2	(28619D)	(U861)	0	(D3)
4	(28619D)	(6861)	0	(D3)
5	(28619D)	(2861)	1	(D3)

Рис 3.8 Результат роботи декодера із двома помилками в різних модулях вхідної комбінації модульних бітів 2861 і контрольних бітів 9D

ВИСНОВОК

В роботі на отримання кваліфікаційно-освітнього рівня бакалавр ком-п'ютерної інженерії розроблено пристрій корекції модульних помилок в ЗП.

В процесі виконання роботи був змодельованай і протестована модель, яка підтримує роздільне кодування і декодування інформаційних слів довжиною до 16 розрядів і дозволяє виправити помилку в будь-якому чотирьохбітному модулі слова. Також ця модель може бути використана при аналізі роздільного кодування і декодування слів довжиною до 64 розрядів.

В ході роботи розроблена програмна HDL-модель пристрою корекції модульних помилок. Вона може бути використана для імплементації розробки у вигляді ВІС для того, аби мати широке використання в безліч сучасних обчислювальних систем.

В результаті виконаної роботи показано, що використання коригувальних кодів для підвищення надійності ЗП є ефективним засобом..

ЛІТЕРАТУРА

1. Хетагуров Я.А., Руднев Ю.П. Повышение надежности цифровых устройств методами избыточного кодирования. - Москва: Энергия, 1974. 272 с.
2. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки. - М.: Мир, 1976. 594 с.
3. Язык описания аппаратуры AHDL: URL: https://studbooks.net/2070057/informatika/yazyk_opisaniya_apparatury_ahdl (Дата звернення: 15.04.2020)
4. Мак-Вильямс Ф. Дж., Слоэн Дж.А. Теория кодов, исправляющих ошибки. -М.: Связь, 1979. 744 с.
5. Березюк Н.Т., “Кодирование информации. Двоичные коды” Харьков, Выща школа, 1978. - 252 с.
6. Конопелько В.К., Лосев В.В., Надежное хранение информации в полупроводниковых запоминающих устройствах. - М.: Радио и связь, 1986. 240 с.
7. Валуйский В.Н., Самофалов К.Г., Романкевич А.М., Каневский Ю.С., Пиневич М.М.: Прикладная теория цифровых автоматов. – Київ. “Вища школа”, 1987. 375с., іл.
8. Пацюра И.В., Корнейчук В.И., Довбыш Л.В. Надежность электронных систем. К., Світ. 1997. 128 с.
9. National Semiconductor, Application Note 302, Charles Carinalli, Mike Evans, Feb. 1995
10. Стешенко В.Б. Плис фирмы Altera: Элементная база, система проектирования и языки описания аппаратуры 3-Е Изд. 2007. 574с.
11. Гордонов А.Ю., Бекин Н.В., Цыркин В.В. “Большие интегральные схемы запоминающих устройств: Справочник”. - М.: Радио и связь, 1990. 288с.
12. Антонов А.П. Язык описания цифровых устройств AlteraHDL. М.: РадиоСофт, 2001. — 224